SIEMENS

SIMATIC

STEP 7 V5.4 编程

使用手册

2006年03月

安全指南

本手册包括应该遵守的注意事项，以保证您个人的生命安全及财产损失。有关人身安全的注意事项在本手册中均采用安全警示标志加以突出强调，有关财产安全的注意事项并没有安全警示标志，并根据危险等级注明如下：



危险（Danger）
表示若不采取适当的预防措施，将造成死亡、严重的人身伤害或重大的财产损失。



警告（Warning）
表示若不采取适当的预防措施，将可能造成死亡、严重的人身伤害或重大的财产损失。



小心（Caution）
有警示标志表示若不采取适当的预防措施，将可能造成轻微的人身伤害或财产损失。

小心（Caution）
无警示标志，表示若不采取适当的预防措施，将可能造成财产损失。

注意（Note）
提醒你对与产品有关的重要信息、产品的处置或文件的特别部分，应格外注意。

如果出现不是同级的警示信息，则须采取最高一级警示。对于人身伤害的警示一般包括对于财产损失的警示。

合格人员

只有合格人员才允许安装和操作这一设备。合格人员规定为根据既定的安全惯例和标准批准进行试运行、接地和为电路、设备和系统加装标签的人员。

正确使用

注意如下：



警告
本装置及其组件只能用于产品目录或技术说明书中阐述的应用，并且只能与西门子公司认可或推荐的其它生产厂的装置或组件相连接。
本产品只有在正确的运输、贮存、组装和安装的情况下，按建议方式进行运行和维护，才能正确而安全地发挥其功能。

商标

SIMATIC®、SIMATIC HMI®和SIMATIC NET®为西门子公司的注册商标。
本手册中所及其它名称也可能是注册商标，禁止未经允许为第三方所使用。

郑重声明
我们已核对过，本手册的内容与所述硬件和软件相符。但错误在所难免，不能保证完全的一致。本手册中的内容将定期审查，并在下一版中进行修正。欢迎提出改进意见。

邮政信箱4848，纽伦堡D- 90437

西门子股份有限公司

前言

目的

本手册详细阐述了如何使用STEP 7进行编程，为您在安装和调试软件时提供支持。本手册解释了如何创建程序，并对用户程序组件作了说明。

本手册的适用对象是那些使用STEP 7和SIMATIC S7自动化系统实现控制任务的人员。

我们建议您首先通过手册《STEP 7 V5.4使用入门》中的例子来了解STEP 7。这些例子让您简单的了解到如何“使用STEP 7进行编程”。

所需基本知识

要了解本手册，需要具有自动化技术的常规知识。

另外，还应熟悉安装有MS Windows 2000 Professional或MS Windows XP Professional或MS Windows 2003 Server 等操作系统的计算机或PC一类的工具的使用（例如，编程器等）。

手册的应用范围

本手册适用于STEP 7 V5.4版本编程软件包。

关于最新的service pack的信息可以在下列文档中找到：

- 在“readme.wir”文件中
- 在更新的STEP 7在线帮助中

在在线帮助中的“What’s new?（新增内容）”主题中可以得到详细介绍，以及STEP 7新增功能。

在线帮助

集成在软件中的在线帮助是本手册的补充。在线帮助的目的是为你提供详细的软件使用帮助。

帮助系统通过多个界面集成在软件中：

- 在Help菜单中有多个菜单命令可以选择：使用“Contents（内容）”命令，可以打开Step 7的帮助索引。
- Using Help（使用帮助）提供有详细的在线帮助使用说明。
- 上下文相关帮助可以提供关于当前的文本信息，例如，一个打开的对话框或一个激活的窗口。你可以通过点击“Help”按钮或按动F1，打开文本相关的帮助。
- 状态栏提供有其它形式的上下文相关帮助。当鼠标放在某个菜单命令上时，它为每个菜单命令显示一个简短的解释。
- 当鼠标短时放在一个工具栏的图标上时，也能为每个图标显示一个简短的解释。

如果你更愿意阅读打印出来的在线帮助，你可以打印每个帮助主题、工作簿或整个在线帮助。

本手册是从基于HTML的STEP 7帮助中摘取出来的。详细过程请参阅STEP 7帮助。由于手册和在线帮助的结构几乎一样，所以能够很容易地在手册和在线帮助之间进行转换。

其它帮助

如果您有任何技术问题，请与当地Siemens代表处联系。

http://www.siemens.com/automation/partner

SIMATIC培训中心

西门子公司还提供有许多培训课程，介绍SIMATIC S7自动化系统。详情请与您所在地区的培训中心联系，或与德国纽伦堡（邮编D-90327）的总部培训中心联系：

电话：　+8610　64721888

电话：　+49 (911) 895-3200

或与当地西门子培训中心联系：

北京：010 – 84597588

上海：021 – 62815933 – 305

广州：020 – 37619450

武汉：027 – 85486688 – 6400

沈阳：0451 – 22949880

重庆：023 – 63828919 – 3002

http://www.ad.siemens.com.cn/training

http://www.sitrain.com

SIMATIC客户支持热线

昼夜值班，遍布全球：



面向全球（纽伦堡） 技术支持 提供24小时服务 电话：+49(0) 180 5050-222 传真：+49(0) 180 5050-223 E-Mail：adsupport@siemens.com GMT：+1:00		
欧洲/非洲（纽伦堡） 授权 当地时间：星期一—星期五 8:00 至17:00 电话：+49(0) 180 5050-222 传真：+49(0) 180 5050-223 E-Mail：adsupport@siemens.com GMT：+1:00	**美国（约翰逊市）** 技术支持和授权 当地时间：星期一—星期五 8:00 至17:00 电话：+1(0) 770 740 3505 传真：+1(0) 770 740 3699 E-Mail： isd-callcenter@sea.siemens.com GMT：-5:00	**亚洲/澳大利亚（北京）** 技术支持和授权 当地时间：星期一—星期五 8:30 至17:30 电话：+86 10 6475 7575 传真：+86 10 6474 7474 E-Mail： adsupport.asia@siemens.com GMT：+8:00
SIMATIC热线和授权热线使用的语言是德语和英语。		

SIMATIC客户支持在线服务

SIMATIC客户服务支持部门，通过其在线服务，还可为您提供与更丰富的有关SIMATIC产品的其它信息：

http://www.siemens.com/automation/service&support

在此，您可以得到：

- 通过新闻向您提供最新的产品信息
- 通过Search功能在Service&Support内查找相应的资料
- 全世界的拥护和专家可以通过论坛交流经验
- 通过数据库可以得到当地自动化与驱动集团的代表处信息
- 在“Services”下得到有关现场服务、维修、备品备件以及更多的信息

目录

1 产品介绍和软件安装

1.1 STEP 7 概述

何谓STEP 7 ？

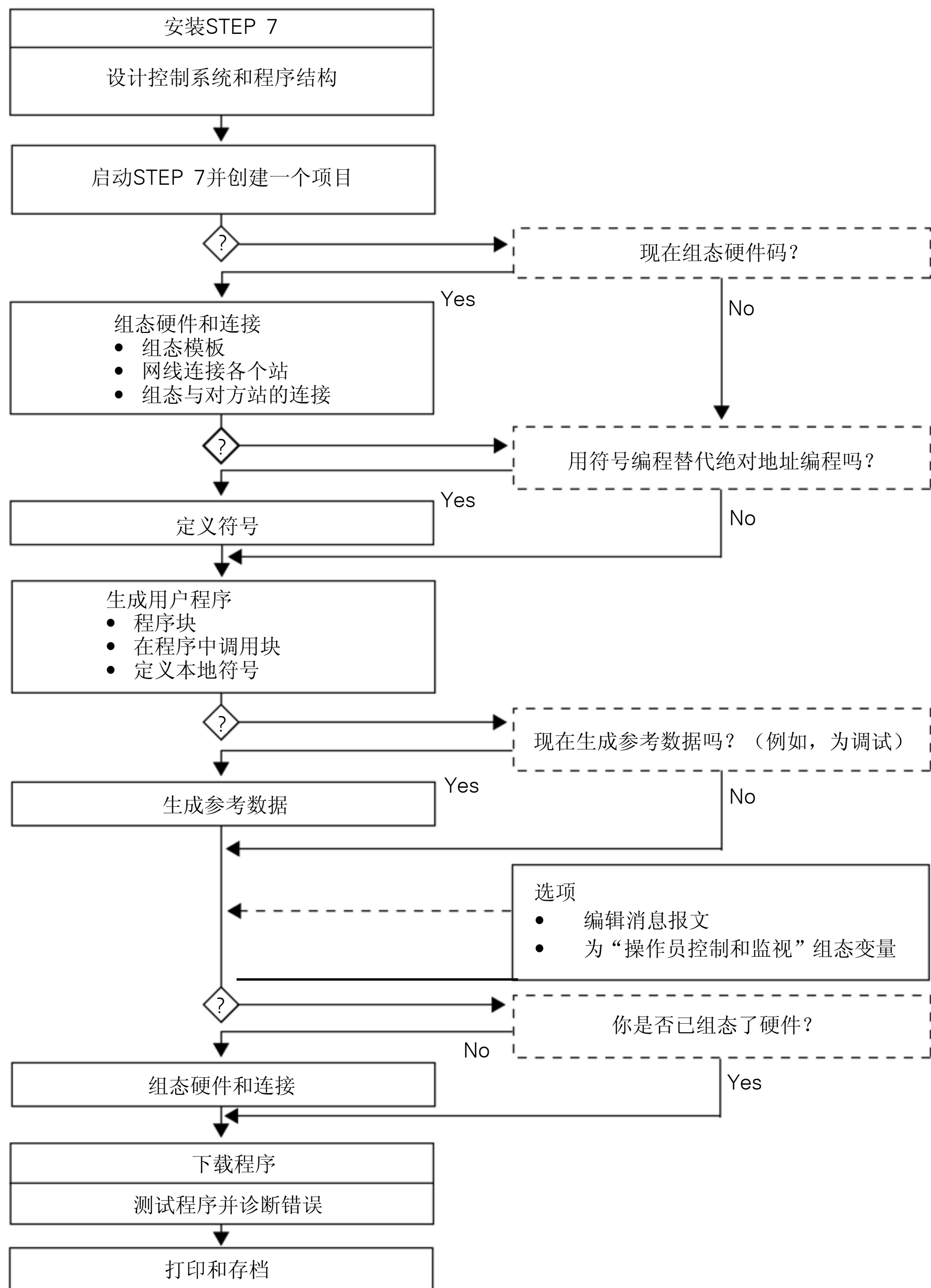
STEP 7是用于对SIMATIC可编程逻辑控制器进行组态和编程的标准软件包。它是SIMATIC工业软件的组成部分。可提供下列STEP 7标准软件包：

- 用于SIMATIC S7-200上应用的STEP 7 Micro/DOS和STEP 7 Micro/WIN。
- 用于使用带有各种功能的SIMATIC S7-300/S7-400、SIMATIC M7-300/M7-400和SIMATIC C 7的STEP 7：
  - 可通过选择SIMATIC工业软件中的软件产品进行扩展（见“STEP 7标准软件包的扩展应用”一节）
  - 为功能模板和通讯处理器赋值参数
  - 强制和多处理器模式
  - 全局数据通讯
  - 使用通讯功能块的事件驱动数据传送
  - 组态连接

STEP 7是本用户手册讨论的主题，STEP 7 Micro在《STEP 7 Micro/DOS用户手册》中描述。

基本任务

当你用STEP 7创建一个自动化解决方案时，有一系列的基本任务。下图所示为大多数项目需要执行的任务，并把这些任务分配到基本程序中。它会提示你与之相关的章节，因此你可以很方便地在整个手册中查找与任务相关的信息。



安装STEP 7
设计控制系统和程序结构
启动STEP 7并创建一个项目
?
现在组态硬件码？
Yes
No
组态硬件和连接
• 组态模板
• 网线连接各个站
• 组态与对方站的连接
?
用符号编程替代绝对地址编程吗？
Yes
No
定义符号
生成用户程序
• 程序块
• 在程序中调用块
• 定义本地符号
?
现在生成参考数据吗？（例如，为调试）
Yes
No
生成参考数据
选项
• 编辑消息报文
• 为“操作员控制和监视”组态变量
?
你是否已组态了硬件？
No
Yes
组态硬件和连接
下载程序
测试程序并诊断错误
打印和存档

可选步骤

如上图所示，你有两个可选步骤：

- 先组态硬件然后编制程序块。
- 也可以先编程序块而不组态硬件。这种方法建议用于维修和维护工作，例如，整合所编制的程序块到一个已有的项目中。

各个步骤的简要说明

- 安装及授权
  第一次使用STEP 7时，要进行安装并将授权从磁盘传至硬盘（见“安装STEP 7”和“授权”一节）。
- 设计你的控制系统
  在使用STEP 7之前，设计你的自动化解决方案。将过程分割为单个的任务以生成一个组态图表（见“设计一个自动化项目的基本步骤”一节）。
- 设计程序结构
  使用STEP 7中可用的块将你的控制系统设计草案中所描述的任务转化为程序结构（见“用户程序中的块”一节）。
- 启动STEP 7
  从Windows用户界面启动STEP 7（见“启动STEP 7”一节）。
- 创建一个项目结构
  项目就像一个文件夹，所有数据都以分层的结构存于其中，任何时候你都可以使用。在创建了一个项目之后，所有其它任务都在这个项目下执行（见“项目结构”一节）。
- 组态一个站
  组态一个站时须指定你要使用的可编程控制器；例如，SIMATIC 300、SIMATIC 400、SIMATIC S5（见“插入站”一节）。
- 组态硬件
  组态硬件就是在组态表中指定你的自动化解决方案所要使用的模板以及在用户程序中以什么样的地址来访问这些模板。模板的特性也可以用参数进行赋值（见“组态硬件的基本步骤”一节）。
- 组态网络和通讯连接
  通讯的基础是预先组态网络。为此，要创建一个你的自动化网络所需要的子网，设置子网特性，设置网络的连接特性以及所有连网的站所需要的通讯连接（见“组态子网的步骤”一节）。
- 定义符号
  可以在符号表中定义本地或全局符号，并在你的用户程序中用这些更具描述性的符号名替代绝对地址。（见“生成一个符号表”一节）。
- 创建程序
  用一种可供使用的编程语言创建一个与模板相连接或与模板无关的程序并以块、源文件或图表的形式存储（见“生成逻辑块的基本步骤”和“编写STL源文件的基本信息”两节）。

- 只适于S7：生成并评估参考数据
  利用这些参考数据可以使用户程序的调试和修改更容易（见“可用参考数据概述”一章）。
- 组态消息报文
  你可以生成与块相关的消息报文，比如包括它们文本内容和属性。并将生成的报文组态数据送至操作员系统数据库（例如SIMATIC WinCC。SIMATIC ProTool），（见“组态报文”一章）。
- 组态操作员控制和监视的变量
  在STEP 7中可以生成操作员控制和监视的变量并赋予它们所需的属性。并将它们传至操作员系统（如WinCC）的数据库（见“组态操作员控制和监视变量”一节）。
- 下载程序到可编程控制器
  只适于S7：完成所有的组态、参数赋值和编程任务之后，可以下载整个用户程序或其中的单个块到可编程控制器（你的硬件解决方案的可编程模板）。（见“下载条件”一节）。CPU已包含操作系统。
  只适于M7：从众多不同的操作系统中为你的自动化解决方案选择一个合适的操作系统，将该操作系统单独地或与用户程序一起传至M7可编程控制系统的存储区中。
- 测试程序
  只适于S7：进行测试，可以显示来自你的用户程序的变量数值，也可以是来自CPU的。对变量可以作赋值，为要显示或修改的变量生成一个变量表（见“用变量表进行测试的介绍”一节）。
  只适于M7：使用高级语言测试工具，测试用户程序。
- 监视操作，诊断硬件
  通过显示模板的在线信息，可以断定模板故障。在诊断缓存区和堆栈内容的帮助下，可以判定用户程序处理中错误的原因。还可以检查一个用户程序是否可以在一个特定的CPU上运行（见“诊断硬件”和“显示模板信息”两节）。
- 制作设备文档
  在创建一个项目/设备后，为项目数据生成清楚的文档资料是非常有意义的，它使该项目的进一步编辑以及维护操作更容易（见“打印项目文档”一节）。DOCPRO是用来创建和管理设备文档的可选工具，使用该工具，你可以设计项目数据结构，译成接线手册的形式，并以通用格式打印出来。

特殊主题

当你创建一个自动化解决方案时，有些特殊主题可能会令你感兴趣。

- 多处理方式—多CPU的同步操作（见“多处理方式—多CPU的同步操作”一节）
- 多用户工作在一个项目中（见“多用户编辑项目”一节）
- 工作在M7系统（见“M7系统程序的方法”一节）

1.2 STEP 7 标准软件包

使用标准

STEP 7中集成的SIMATIC编程语言和语言表达方式，符合EN 61131-3标准。标准软件包符合图形化以及面向对象的Windows操作系统要求，可以运行在操作系统Windows 2000/XP专业版以及Windows Server 2003下。

标准软件包的功能

标准软件支持自动任务创建过程的各个阶段，如：

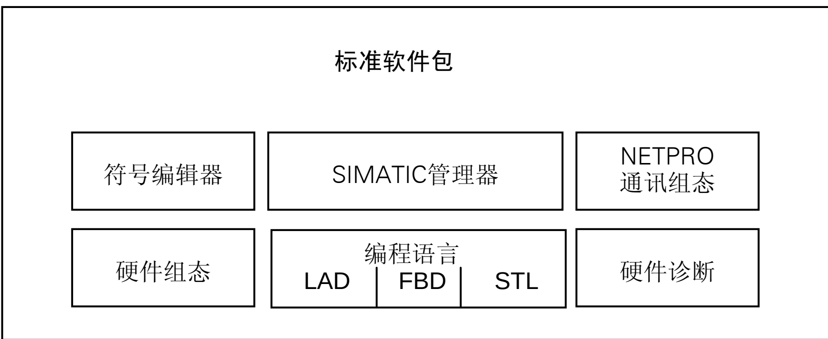
- 建立和管理项目
- 对硬件和通讯作组态和参数赋值
- 管理符号
- 创建程序，例如为S7可编程控制器创建程序
- 下载程序到可编程控制器
- 测试自动化系统
- 诊断设备故障

STEP 7软件的用户接口，基于当前最新水平的人机控制工程设计，轻松使用。

STEP 7软件产品手册在在线帮助和PDF格式电子手册中提供有所有在线信息。

STEP 7中的应用程序

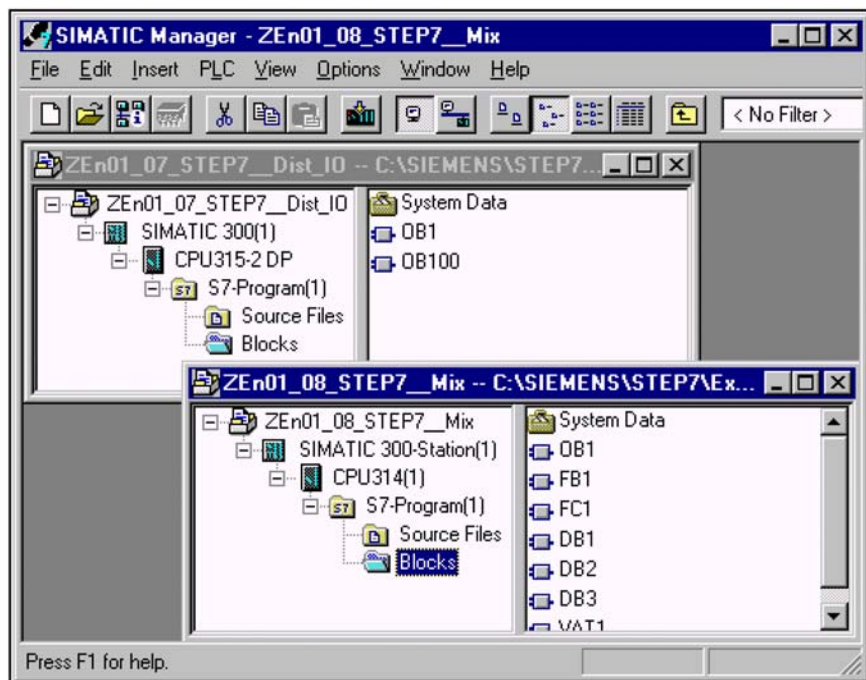
STEP 7标准软件包提供一系列的应用程序（工具）：



无需分别打开各个工具；当选择相应功能或打开一个对象时，它们会自动启动。

SIMATIC 管理器

SIMATIC Manager（SIMATIC管理器）可以管理一个自动化项目的所有数据，无论是为哪个可编程控制系统（S7/M7/C7）设计的。编辑所选数据的工具由SIMATIC Manager自行启动。



符号编辑器

使用Symbol Editor（符号编辑器），可以管理所有的共享符号。具有以下功能：

- 为过程信号（输入/输出）、位存储和块设定符号名和注释
- 分类功能
- 从/向其它的Windows程序导入/导出

使用这个工具生成的符号表可供其它所有工具使用。因而，对一个符号属性的任何变化都能自动被其它工具识别。

诊断硬件

这些功能可以向你提供可编程控制器的状态概况。这个概况中可以显示符号，指示每个模板是否正常或有故障。双击故障模板，可以显示有关故障的详细信息。信息的范围视各个模板而定：

- 显示关于模板的一般信息（例如，订货号、版本、名称）以及模板状态（例如，故障）
- 显示中央I/O和分布式（DP）从站的模板信息（例如，通道故障）
- 显示来自诊断缓存区的消息报文

对于CPU，还可显示以下附加信息：

- 用户程序处理过程中的故障原因
- 显示循环时间（最长的、最短的和最近一次的）
- MPI的通讯可能性及负载
- 显示性能数据（可能的输入/输出、位存储、计数器、定时器和块的数量）

编程语言

用于S7-300和S7-400的编程语言梯形逻辑图（Ladder Logic）、语句表（Statement List）和功能块图（Function Block Diagram）都集成在一个标准软件包中。

- 梯形逻辑图（或LAD）是STEP 7编程语言的图形表达方式。它的指令语法与一个继电器的梯形逻辑图相似：当电信号通过各个触点、复合元件以及输出线圈时，使用梯形图，可以追踪电信号在电源示意线之间的流动。
- 语句表（或STL）是STEP 7编程语言的文本表达方式，与机器码相似。如果一个程序是用语句表编写的，CPU执行程序时则按每一条指令一步一步地执行。为使编程更容易，语句表已进行扩展，还包括一些高层语言结构（例如，结构数据的访问和块参数）。
- 功能块图（FBD）是STEP 7编程语言的图形表达方式，使用与布尔代数相类似的逻辑框来表达逻辑。复合功能（如数学功能）可用逻辑框相连直接表达。

其它编程语言作为可选软件包使用。

硬件组态

使用这个功能，可以为自动化项目的硬件进行组态和参数赋值。具有以下功能：

- 组态可编程控制器时，可以从电子目录中选择一个机架，并在机架中将选中的模板安排在所需要的槽上。
- 组态分布式I/O与组态中央I/O一致，也支持以通道为单位的I/O。
- 在给CPU赋值参数的过程中，可以通过菜单的指导设置属性，比如，启动特性和循环扫描时间监控。支持多处理方式。输入的数据保存在系统数据块中。
- 在向模板作参数赋值过程中，所有可以设置的参数都是用对话框来设置的。没有任何设置使用DIP开关。向模板的参数赋值传送是在CPU启动过程中自动完成的。这意味着，例如，模板可以调换而无需赋值新的参数。
- 功能模板（FM）和通讯处理器（CP）的参数赋值，与其它模板的赋值方法一样，也是在硬件组态工具中完成的。对于每一个FM和CP，都有模板特定对话框和规则（包括在FM/CP功能软件包范围内）。通过只在对话框中提供有效的选项，系统可以防止不正确的参数输入。

NetPro（网络组态）

通过MPI，NetPro可以实现时间驱动的循环数据传送：

- 选择通讯的站点。
- 在表中输入数据源和数据目标；自动生成要下载的所有块（SDB），并且自动完整地下载到所有的CPU中。

事件驱动的数据传送也是可以实现的：

- 设置通讯连接。
- 从集成的功能块库文件中选择通讯或功能块。
- 根据你选择的编程语言为所选的通讯或功能块赋值参数。

1.3 STEP 7 V5.4 中的新内容

STEP 7 V5.4更新了以下内容：

- SIMATIC 管理器
- 组态和硬件诊断
- 网络组态和连接
- 标准库
- 系统错误报告

SIMATIC 管理器

- 在STEP V5.4中有两种时间和日期的格式：STEP 7国际通用格式以及ISO 8601标准格式。设置时，进入SIMATIC 管理器，打开“Customize”对话框选择“Date and Time”标签。
- 在STEP V5.4中在PG/PC的本地时间显示中可以显示模板时间。设置时，进入SIMATIC 管理器，打开“Customize”对话框选择“Date and Time”标签。
- 在STEP 7 V5.4中，可以通过设置密码来限制对项目及库文件的访问。设置时，须首先安装 SIMATIC Logon V1.3 SP1（请参照SIMATIC logon 如何进行访问） 。
- 在STEP 7 V5.4中设置了项目访问保护后，还可以保持日志的在线记录，比如“Download”，“Operating Mode Changes”和“Reset”。设置时，须首先安装SIMATIC Logon V1.3 SP1（请参照SIMATIC logon 如何进行访问保护）。

组态和诊断硬件

- 支持“Information and Maintenance”功能，可以读/写模板识别信息。（参考I&M）
- 在冗余模式下，模板的识别信息可以被写入到PROFIBUS DP的接口模板中（通过“Accessible Nodes”）。该接口模板须支持该功能。
- C A x 数据能够被导入/导出。这样，STEP 7 与 CAD 或 CAE 等工程工具系统之间可以方便的进行数据交换（请参考导入/导出C A x数据）。
- PROFIBUS DP接口模块可以在冗余模式下进行硬件版本的升级，只要接口模板支持该功能。另外，通过有源背板，冗余的接口模板也可以将更新的硬件版本发送给其他冗余的接口模板。
- 软件冗余“Software Redundancy”功能现在允许PA从站在冗余系统中被复制。（请参考组态SW Redundancy）。
- 编辑硬件组态现在可以通过“Edit > Open Object ”菜单指令直接打开(请参考Opening Object in HW Config）。
- 可以为PROFINET IO设备组态看门狗时间（Watchdog Time）（请参考Configuring the Watchdog Time）。

组态网络和连接

- 支持PROFINET IO IRT通讯（等时实时通讯 Isochronous Realtime）。这意味着短时的等时总线循环时间可以被组态。（请参考Isochronous Realtime Ethernet）
- 可以直接将IO设备复制到另外的站中。如果IP地址重复，可以定义在插入时修改（保留IP地址或重新分配一个地址）。
- 可以为PROFINET IO设备分配看门狗时间（Watchdog Time），在“IO Cycle”中，该选项在设备属性中可以选择。
- 当PROFIBUS DP中使用光纤时，特别是当组态了光纤环网时，可以定义OLM的个数。这将有助于总线参数的计算更为精确。另外，也意味着使用高性能的设备后，总线时间将被缩短。

标准库

- 标准库“Communication Blocks(通讯块)”中扩展了FB67 和FB68 用于Open TCP/IP通讯。
- 标准库“Communication Blocks(通讯块)”中扩展了FB20、FB21、FB22和FB23用于循环读取标准PROFIBUS 用户设备的数据(PROFIBUS Nutzerorganisation e.V PNO)。
- 除了已经存在的冗余库“Redundant IO（V1）”，STEP 7 V 5.4 加入了新库“Redundant IO CGP”（channel granular peripheral devices）。支持单模板的通道冗余。请参考STEP 7 的readme 文件中关于该部分的描述，以及相关FAQ：http://support.automation.siemens.com/

系统错误报告

- 支持PROFIBUS诊断功能数据块DB125，从而可以将诊断数据发送到HMI设备上。

1.4 STEP 7 标准软件包的扩展应用

标准软件包可通过可选软件包进行扩展，这些可选软件包被分组为以下三个软件类别：

- 工程工具（Engineering Tool）；这些是较高层次的编程语言和面向工艺的软件。
- 运行版软件（Run-Time Software）； 这是用于生产过程的而无需架框的运行版软件。
- 人机接口（Human Machine Interface，HMI）；这是特别用于操作员控制和监视的软件。

下表是可选软件，你可以根据你的可编程控制系统来选用：

	STEP 7		
	S7-300 S7-400	M7-300 M7-400	C7-620
Engineering Tool			
• Borland C/C++		o	
• CFC	+[1]	+	+[2]
• DOCPRO	+	+[3]	+
• HARDPRO	+		
• M7 ProC/C++		o	
• S7 GRAPH	+[1]		+[2]
• S7 HiGraph	+		+
• S7 PDIAG	+		
• S7 PLCSIM	+		+
• S7 SCL	+		+
• Teleservice	+	+	+
Run-Time Software			
• Fuzzy Control	+		+
• M7-DDE Server		+	
• M7-SYS RT		o	
• Modular PID Control	+		+
• PC-DDE Server	+		
• PRODAVE MPI	+		
• Standard PID Control	+		+
Human Machine Interface			
• ProAgent			
• SIMATIC ProTool			
• SIMATIC ProTool/Lite			o
• SIMATIC WinCC			

O = 必须的

+ = 可选的

1）= 从S7-400以上建议使用

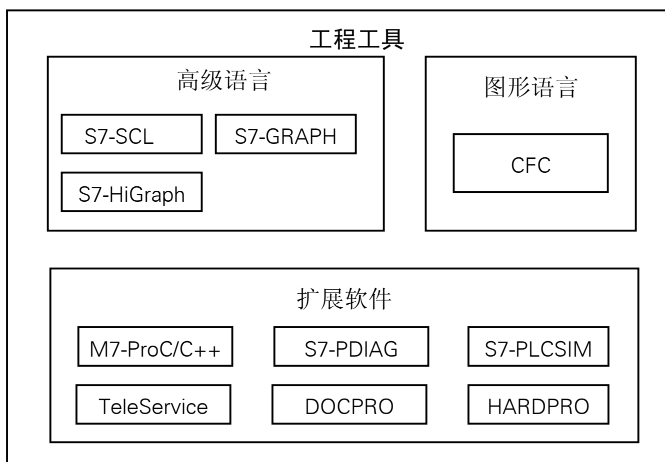
2）= 对于C7-620不推荐

3）= 不用于C程序

1.4.1 工程工具（Engineering Tool）

工程工具（Engineering Tool）是面向任务的工具，可被用于扩展标准软件包。工程工具（Engineering Tool）包括：

- 供编程人员使用的高级语言。
- 供技术人员使用的图形语言。
- 用于诊断、模拟、远程维护、设备文档制作等的扩展软件。



高级语言

下列语言作为可选软件包，用于SIMATIC S7-300/S7-400可编程控制器的编程：

- S7 GRAPH是用于编制顺序控制（步和转换）的编程语言。在这种语言中，过程顺序被分割为步。步中包含控制输出的动作。从一步到另一步的转换由转换条件控制。
- S7 HiGraph是以状态图的形式描述异步、非顺序过程的编程语言。为此，系统要被分解为几个功能单元，每个单元呈现不同的状态。各功能单元可通过在图形之间交换报文来同步。
- S7 SCL是符合EN 61131-3（IEC 1131-3）标准的高级文本语言。它包含的语言结构与编程语言Pascal和C相类似。所以S7 SCL特别适合于习惯于使用高级编程语言的人使用。例如，S7 SCL可以用于编程复杂或经常重复使用的功能。

图形语言

CFC是用于S7和M7的编程语言，用来以图形方式连接已有的功能。这些功能涵盖了从简单的逻辑操作到复杂的闭环和开环控制等极为广泛的功能范围。大量的此种类型的功能在库中以块的形式提供。编程时将这些块拷贝到图表中并用线连接。

扩展软件

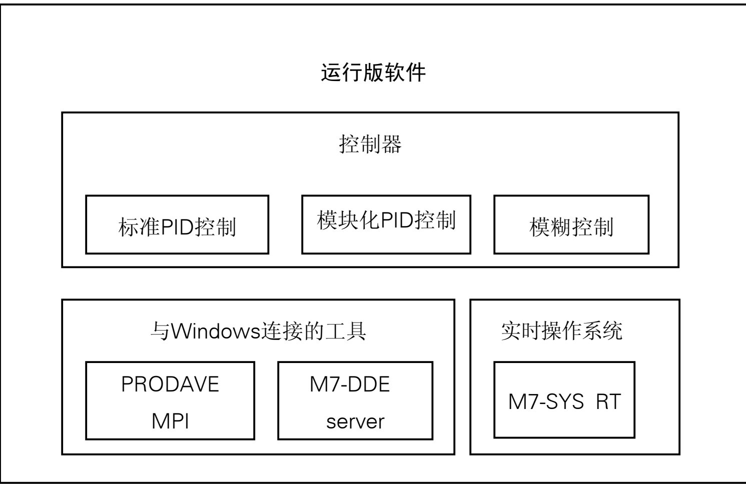
- Borland C++（只用于M7）包含Borland开发环境。

- 使用DOCPRO，可以将你用STEP 7生成的全部组态数据构造为接线手册。这使得组态数据的管理更为容易，并且可以为按照指定标准的打印准备好信息。
- HARDPRO是S7-300硬件组态系统，它支持用户对复杂的自动化任务的大范围的组态。
- M7-ProC/C++（只用于M7），允许将编程语言C和C++的Borland开发环境集成到STEP 7的开发环境中。
- 你可以使用S7 PLCSIM（只用于S7）模拟将S7可编程控制器连接到编程器或PC，以便进行测试。
- 使用S7 PDIAG（只用于S7），可以标准化组态SIMATIC S7-300/S7-400过程诊断。使用过程诊断，可以检测可编程控制器之外的故障和故障状态（例如，未到达限位开关）。
- 使用TeleService，就可以使用编程器或PC，通过电话网对S7和M7可编程控制器作远程在线编程和服务。

1.4.2 运行版软件

运行版软件包括可以由用户程序调用的预编程的解决方案。运行版软件直接集成在自动化解决方案中。它包括：

- SIMATIC S7控制器，例如，标准模板，以及模糊控制。
- 用于连接可编程控制器和Windows应用程序的工具。
- SIMATIC M7的一个实时操作系统。



SIMATIC S7控制器

- 使用标准PID控制，可以将连续控制器、脉冲控制器以及步进控制器集成到用户程序中。使用带有集成控制器设置的参数赋值工具，可以让你在一个很短的时间内将控制器设为最优使用。
- 如果简单的PID控制器不足以解决你的自动化任务，则可以使用模块化PID控制。通过

在提供的标准功能块中作连接，几乎可以设计和建立任何控制结构。

- 使用模糊控制可以生成模糊逻辑系统。当过程很难或无法用数学模型来描述；过程特性不可预知；或者出现非线性；但具有过程运行的经验时，可以使用这种模糊系统。

用于连接Windows的工具

- PRODAVE MPI是用于SIMATIC S7、SIMATIC M7和SIMATIC C7之间过程数据通信的工具箱。它通过多点接口（MPI）自行管理数据通信。
- 使用M7-DDE服务器（动态数据交换），无需另外编程就可将Windows应用程序连接到SIMATIC M7的过程变量。

实时操作系统

- M7-SYS RT包含M7 RMOS 32操作系统和系统程序。它是SIMATIC M7软件包使用M7-ProC/C++和CFC的前提条件。

1.4.3 人机接口

人机接口（HMI）是专门用于SIMATIC中操作员控制和监视的软件。

- 开放的过程监视系统SIMATIC WinCC，是一个基本的操作员接口系统，它包括所有重要的操作员控制和监视功能，这些功能可以用于任何工业系统和使用任何工艺。
- SIMATIC ProTool和SIMATIC ProTool/Lite是用于组态SIMATIC操作员面板（OP）和SIMATIC C7紧凑型设备的现代工具。
- ProAgent通过建立有关故障原因和位置的信息可实现对系统和设备的有目的快速过程诊断。



2 安装

2.1 自动化授权管理

2.1.1 通过自动化授权管理器获得用户权

自动化授权管理器

为了使用STEP 7 V5.3编程软件，你需要一个产品特定的授权钥匙(license key)。启动STEP 7 V5.4后，自动化授权管理器将安装该钥匙。

自动化授权管理器是Siemens AG开发的一个软件产品。它可以管理所有自动化系统的授权。它位于：

- 安装在需要一个授权钥匙的软件产品的安装设备上。
- 安装在一个单独的安装设备上。
- 可以从西门子公司的自动化与驱动集团的客户支持网页上下载。

自动化授权管理器具有自己的在线帮助。安装授权管理器后按F1键或选择**Help > Help on License Manager**可以获得帮助。在线包含了有关自动化授权管理器的详细的功能和操作信息。

授权

授权需要使用STEP 7编程软件包，通过授权可以保护您合法使用软件。授权通过CoL(授权认证)和授权钥匙进行保护。

授权认证(CoL)

与产品结合在一起的“授权认证”可以保证用户合法使用该产品的权利。授权认证方以及授权使用方均可使用该产品。

授权钥匙

授权钥匙只是通过技术方法使用户可以使用授权，类似于一个“电子授权标签”。

SIEMENS AG的所有软件均通过授权钥匙对授权进行保护。当计算机启动后，只有符合申请授权的用户并且授权钥匙经验证是合法的用户才能使用该软件。

注意：

- 可以使用无授权钥匙的标准软件，来熟悉软件的用户接口及其相应功能。
- 但是，STEP 7软件必须是正版软件。
- 如果没有安装授权钥匙，则软件将以一定的时间间隔提醒您安装授权钥匙。

授权钥匙可以在以下类型的存储设备中进行存储和传送：

- 在授权钥匙盘上
- 在本机硬盘上
- 在网络硬盘上

如果软件产品没有可安装的授权，则你需要决定需要哪种授权，并根据需要订购。

获得和使用授权钥匙的进一步信息，请参见自动化授权管理器的在线帮助。

授权类型

SIEMENS AG的软件产品提供多种类型的用户授权，可根据应用要求进行选择。安装不同类型的授权钥匙可以使软件具有不同的实际性能。

授权类型	说明
单用户授权 (Single License)	软件只能在单独的计算机上使用，使用时间不限。
浮动授权 (Floating License)	软件可以在一个计算机网络上使用(“远程使用”)，使用时间不限。
试验授权 (Trial License)	可按照下列限制使用软件： • 最多14天的使用时间 • 从第一次使用后的全部运行天数 • 用于测试使用
租赁授权 (Rental License)	软件使用有如下限制： • 最长50天 • 或按照用户使用的小时数计算使用时间
升级授权 (Upgrade License)	当前系统的特定需求可能需要软件升级： • 可以使用升级授权将老版本的软件升级到新版本的软件 • 由于系统处理数据量的增加可能需要升级

2.1.2 安装自动化授权管理器

通过MSI启动步骤安装自动化授权管理器。在STEP 7的CD盘上有该安装软件。

可以与STEP 7软件同时安装，也可以在安装STEP 7软件后再安装。

注意：

- 安装的详细信息参见“Readme.wri”文件。
- 自动化授权管理器的在线帮助中介绍了授权钥匙的功能以及使用的全部信息。

授权钥匙的后续安装

启动STEP 7软件后如果没有授权钥匙，将显示警告信息。

此时可通过下列方法安装授权钥匙：

- 从软盘安装
- 从网站下载。此时，您必须事先已订购了该授权钥匙
- 在一个网络中使用浮动授权

安装的详细信息，请参见自动化授权管理器的在线帮助。按F1键或选择菜单命令**Help > Help on License Manager**。

注意：

- 在Windows 2000/XP/Server 2003下，只有安装在本机硬盘上并具有写访问状态时，才能进行授权。
- 只能在网络内使用浮动授权(“远程使用”)。

2.1.3 管理授权的原则



注意

阅读本章中和授权盘上的“README.WRI”文件中的提示。如果你不遵循这些准则，可能会丢失授权且无法挽回。

通过F1功能键， 可以快捷的打开在线帮助文件，当然，也可以通过选择Help>Help on License Manager 打开帮助文件。

该帮助文件包括所有的关于授权操作的信息。

2.2 安装 STEP 7

STEP 7可以通过Setup程序自动地进行安装。用户可按照屏幕弹出的指南信息的引导，一步一步地完成整个安装步骤。用户可按照标准的Windows 2000/XP/Server 2003软件安装步骤调用Setup程序。

主要安装步骤：

- 将数据拷贝到你的编程器上
- 设置EPROM和通讯的驱动器
- 安装授权（如果需要）

提示

西门子编程器已将STEP 7软件装在硬盘上。

安装要求

- 操作系统：Windows 2000或Windows XP，Windows Server 2003。
- 基本硬件：

  编程器或PC：

  - 奔腾处理器（600 MHz）
  - RAM：至少256 MB
  - Microsoft Windows 支持的彩色显示器、键盘和鼠标

编程器（PG）是专门为在工业环境中使用而设计的紧凑型个人计算机。它安装了用于SIMATIC可编程序逻辑控制器编程时所需的一切。

- 硬盘空间：请参考readme.wri文件中提供的所需硬盘空间的要求。
- MPI接口（可选）：如果用户需要使用STEP 7与PLC进行通讯，则在PG或PC与PLC之间可使用多点接口（MPI）。

因此，用户需要：

- 一个PC USB 适配器连接到设备的通讯口
- 在计算机中安装MPI模块(例如CP 5611)

PG中已经装有MPI接口。

- 外部编程设备（可选）：如果用户希望在PC上向EPROM中存储程序，则还需要一个外部编程设备。

提示

见README.WRI文件中的“安装STEP 7”的注意事项和“与STEP 7基本软件包兼容的SIMATIC软件包清单”。

使用命令**Start > Simatic > Product Notes**，在“Start（开始）”菜单中，可以找到Readme文件。

使用命令**Start > Simatic > Documentation**，在“Start（开始）”菜单中，可以找到兼容性列表。

2.2.1 安装步骤

安装准备

在用户开始安装软件之前，必须先启动Windows 2000/XP或Server 2003。

- 如果在用户编程器的硬盘上已备份有STEP 7软件，则用户不再需要任何外部数据媒介。
- 从光盘上安装时，要将光盘放入用户PC机的光驱中。

开始安装程序

按如下步骤安装软件：

1. 插入光盘，双击文件“SETUP.EXE”，起动安装程序。

2. 一步一步地按照安装程序所显示的指令进行。

在整个安装过程中，安装程序一步一步地指导用户如何进行。在安装的任何阶段，用户都可以切换到下一步或上一步。

在安装过程中，在对话框中显示有一些选项需要用户选择。请阅读下列提示，可帮助你既快又容易地回答一些安装访问。

如果已经安装了STEP 7的某一种版本

如果安装程序发现，在编程器上已有另一版本的STEP 7，它将报告该情况，并提示用户如何进行：

- 中断安装，以便用户可以将旧的STEP 7版本在Windows下卸载，然后，再开始安装。
- 继续安装，用新版本覆盖旧版本。

如果用户在安装新版本之前，卸载旧版本，则用户软件能够较好地组织软件组件。使用新版本覆盖旧版本有一个缺点是如果以后做旧版本卸载时，旧版本中有些部分不能被完全删除。

选择安装选项

在用户选择安装范围时，有三种选项：

- 标准安装：用于用户接口的所有语言、所有应用以及所有的举例。请参考最新产品信息中对这种组态所要求的存储空间。
- 基本安装：只有一种语言，没有举例。请参考最新产品信息中对这种组态所要求的存储空间。
- 用户定义安装：用户可定义安装范围，例如用户可以选择希望安装的程序、数据库、举例和通讯功能。

ID号码

在安装的过程中，需要用户输入一个ID号码。该号码，可在软件产品证书中或授权盘上找到。

安装授权

在安装过程中，安装程序将检查硬盘上是否有授权。如果没有发现授权，会出现一条信息，指出该软件只能在有授权的情况下使用。如果用户愿意，可立即运行授权程序（此时插入授权盘即可），或者继续安装，稍后再执行授权程序。

PG/PC接口设置

在安装过程中，会出现一个对话框，在这个对话框中，用户可以设置PG/PC接口的参数。用户可在“设置PG/PC接口（Setting the PG/PC Interface）”中找到更多信息。

设置存储卡参数

在安装过程中，会出现一个对话框，在这个对话框中，用户可以为存储卡分配参数。

- 如果用户不用存储卡，则不需要EPROM驱动器.选择“NO EPROM Driver”选项。
- 否则，选择应用到你的编程器上条目。
- 如果用户使用的是PC，则可选择用于外部编程设备的驱动器。这里，用户必须定义哪个接口用于连接外部编程设备（例如，LPT1）。

在安装完成之后，用户可通过STEP 7或控制面板中的“Memory Card Parameter Assignment（存储卡参数赋值）”修改这些参数。

闪存文件系统

在为存储卡赋参数的对话框中，用户可以定义是否应安装闪存文件系统。

例如，当用户需要在SIMATIC M7中向EPROM存贮卡中写入某些文件，或从EPROM存贮卡中删除某些文件，同时，不改变存储卡中保留的内容时，就需要有闪存文件系统。

如果用户使用某个编程器（PG720/PG740/PG760）或用外部编程设备，并且，希望使用此功能，则选择闪存文件系统的安装。

如果在安装过程中出现错误

下列错误可能导致安装失败：

- 如果在起动后即出现一个初始化错误，该程序很可能不能在Windows下起动。
- 没有足够的存储空间：对于标准软件，不考虑用户安装的范围，在硬盘上至少需要100M的空间。
- 坏光盘：如果盘是坏的，请与当地西门子代表处联系。
- 操作员错误：重新安装，并仔细阅读各项指令。

完成安装

如果安装成功，会在屏幕上出条信息告知用户。

如果在安装的过程中，改变了系统文件，建议你重新启动Windows。当用户完成这些以后，可以开始基本的STEP 7应用，即SIMATIC管理器。

一旦安装成功完成，会为STEP 7生成一个程序组。

2.2.2 设置PG/PC接口

用户可以通过这里设置编程器/PC与可编程序控制器之间的通讯连接。在安装过程中，会出现一个对话框，在这个对话框中，用户可以设置PG/PC接口的参数。在安装之后，用户也可以在STEP 7中调用“Setting PG/PC接口”程序，显示该对话框。这将使你在安装之后，可以改变接口参数。

基本程序

为了对接口进行操作，用户需要如下步骤：

- 在操作系统中设置
- 配置适合的接口参数

如果用户使用PC机里的MPI卡或通讯处理器(CP)，则应检查在Windows 中“Control Panel（控制面板）”里的中断和地址设置，以确保没有中断冲突和地址区重叠。

在Windows 2000/XP和Server 2003下，支持ISA总线的MPI-ISA卡不再被操作系统支持。

为了使向编程器/PC接口分配参数容易进行，在显示的对话框中提供一套预先定义的基本参数（接口参数）供用户选择。

为PG/PG接口分配参数

为了设置模板参数，请按照下列步骤要点进行（可在在线帮助中找到详细描述）：

1. 双击“Control Panel（控制面板）”中的“Setting PG/PC Interface（设置PG/PC接口）”。
2. 将“Access Point of Application（应用访问点）”设置为“S7ONLINE”。
3. 在“Interface Parameter set used（所用接口参数集）”的列表中，选择接口所需的参数。如果没有显示你所需要的接口参数，用户必须用“Select（选择）”按钮先安装模板或协议。然后，接口参数会自动生成。在即插即用系统内，无需手工安装即插即用CP (CP 5611 和 CP 5511)的驱动。当在PG/PC上已安装了硬件后，他们可以自动集成到“Setting PG/PC”上。
   - 如果用户所选的接口能**自动识别总线参数**（例如：CP 5611（Auto）），则用户可以直接将编程器或PC连接至MPI或PROFIBUS上，而不需要设置总线参数。如果传输率小于187.5Kbps，在读总线参数时，有可能有将近1分钟的延迟。**自动识别条件**：主站循环分配总线参数并连接到总线上。所有与此有关的新MPI组件都有此功能；对于PROFIBUS 子网，必须使能该功能（已设为缺省PROFIBUS网络设置）。
   - 如果用户选择的接口无法自动识别网络参数，则可以通过属性设置使其适应总线参数。

如果与其它设置发生冲突，需要做必要的修改（例如，修改中断或地址的设置）。在这种情况下，在Windows 的硬件组态和控制面板中做相应的修改（见下面）。



注意

禁止删除模板参数设置“TCP/IP”（如果“TCP/IP”参数出现的话）。否则可能引起其它应用功能的故障。

检查中断和地址设置

如果用户使用的是PC带有MPI卡，则应检查缺省设置的中断和地址区是否被占用，如果需要，选择一个没被占用的中断和地址区。

在Windows 2000下，用户可以：

- 在Control Panel > Administrative Tools > Computer Management > System Tools > System Information > Hardware Resources下，显示资源设置。
- 在Control Panel > Administrative Tools > Computer Management > System Tools > Device Manager > SIMATIC NET > CP-Name > Properties > Resources下，修改资源。

在Windows XP下，用户可以：

- 在Start > All Programs > Accessories > System > System programs > System Information > Hardware Resources，显示资源设置。
- 在Control Panel > Desktop > Properties > Device Manager > SIMATIC NET > CP Name > Properties > Resources修改资源。

2.3 卸载 STEP 7

使用通常的Windows步骤来卸载STEP 7：

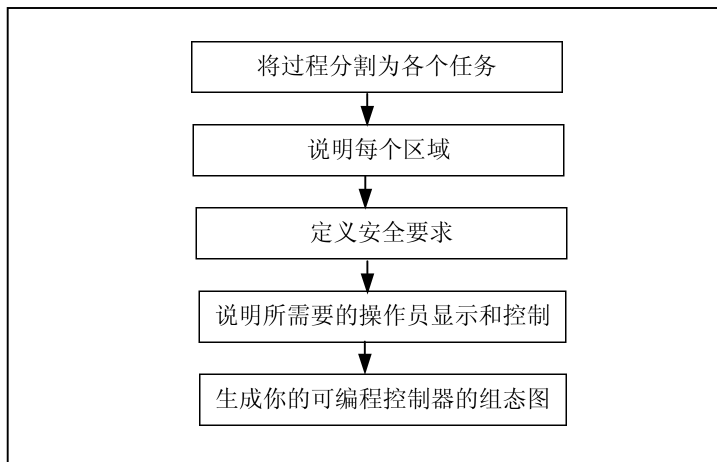
1. 在“Control Panel”中，双击“Add/Remove Programs”图标，启动Windows下用于安装软件的对话框。
2. 在安装软件显示的项目表中，选择STEP 7。点击“Add/Remove（添加/删除软件）”按键。
3. 如果“Remove Shared File（删除共享的文件）”对话框出现，如果用户不能确定是否删除，则可点击“No”按键。

3 设计自动化解决方案

3.1 设计一个自动化项目的基本步骤

本章概述了为一个可编程控制器（PLC）设计一个自动化项目所涉及的基本任务。基于一个自动工业搅拌过程示例的指导，将一步一步贯穿整个过程。

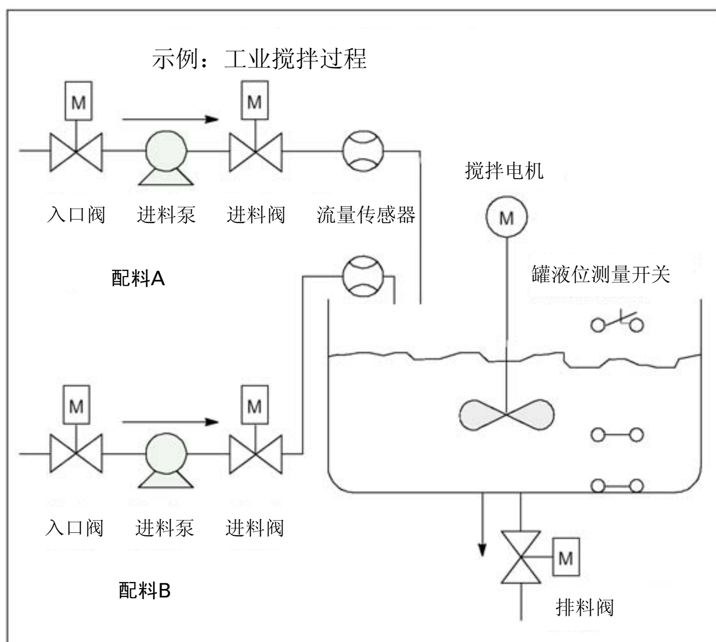
设计一个自动化项目的方法有很多。可用于任何项目的基本步骤的说明如下图所示。



3.2 将过程分割为任务和区域

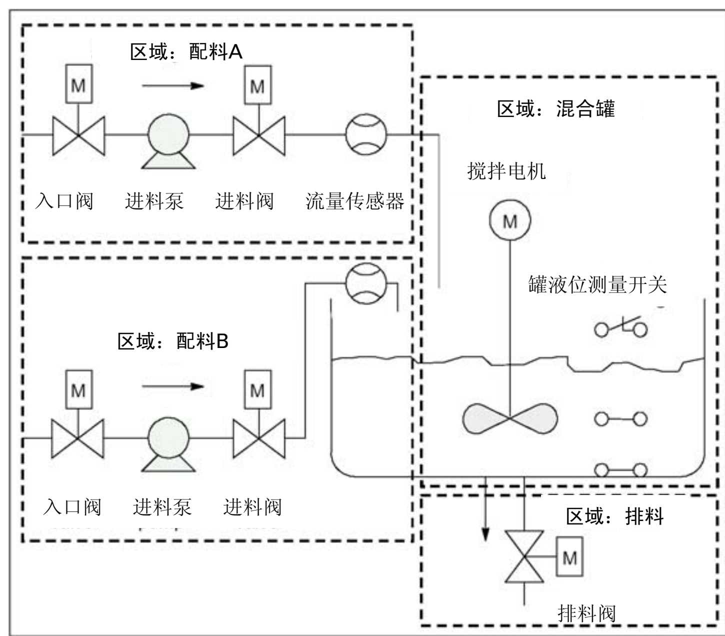
一个自动化过程包括许多单独的任务。通过识别一个过程内的相关任务组，然后将这些组再分解为更小的任务，即使最复杂的过程也能够被定义。

下面这个工业搅拌过程的例子可以用来说明如何将一个过程构造为功能区域和单个的任务：



决定过程的区域

在定义了要控制的过程之后，将项目分割成相关的组或区域：



由于每组被分为小任务，所以控制过程在这一部分所要求的任务就不那么复杂了。

在我们的工业搅拌过程示例中，你可以看到四个不同的区域（见下表）。在这个例子中，配料A的区域中包含的设备与配料B的区域相同。

功能区域	使用的设备
配料A	配料A的进料泵，配料A的入口阀，配料A的进料阀，配料A的流量传感器
配料B	配料B的进料泵，配料B的入口阀，配料B的进料阀，配料B的流量传感器
混合罐	搅拌电机，罐液位测量开关
排料	排料阀

3.3 说明各个功能区域

当你说明过程中的各个区域和任务时，不仅要定义每个区域的操作，而且要定义控制该区域的各种组件。这包括：

- 每个任务的电的、机械的和逻辑的输入和输出
- 各个任务的互锁和相关性

本示例工业搅拌过程使用泵、电机和阀门。必须对这些设备作精确描述，以识别其操作特

性和操作过程所要求的互锁类型。下表提供的示例是对工业搅拌过程中使用的设备的描述。完成说明后，还可以用它来订购所需要的设备。

配料A/B：进料泵电机
进料泵电机传送配料A和B到混合罐 - 流速：400升（100 加仑） /分钟 - 速率：1200 转/分时，100千瓦（134马力）
泵由混合罐附近的操作员站控制（启动/停止）。启动的次数被计数以便进行维护。计数器和显示都可以由一个按钮复位。
对泵进行操作必须满足以下条件： - 混合罐不满 - 混合罐的排料阀关闭 - 紧急关断未动作
如果满足下列条件，则泵被关断： - 在泵电机启动7秒后流量传感器仍指示没有流量 - 流量传感器指示流动已停止

配料A/B：入口阀和进料阀
配料A和B的入口阀和进料阀可以允许或防止配料进入混合槽。
阀门是带有弹簧的螺线管 - 如果螺线管动作则送出阀打开 - 如果螺线管不动作则送出阀关闭
入口阀和进料阀都由用户程序控制。
满足以下条件，排料阀可以打开： - 进料泵电机至少运行1秒
如果满足下列条件，则泵被关断： - 流量传感器指示没有流量

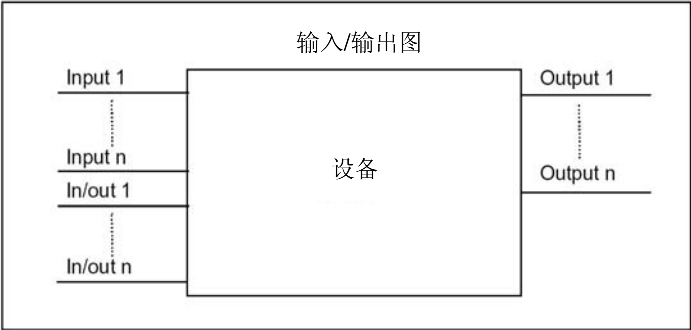
搅拌电机
搅拌电机在混合罐中混合配料A和配料B - 速率：1200 转/分时100千瓦（134马力）
搅拌电机由混合罐附近的操作员站控制（启动/停止）。启动的次数被计数以便进行维护。计数器和显示都可以由一个按钮复位。
对泵进行操作必须满足以下条件： - 罐液位传感器没有指示“罐液位低于最低限” - 混合罐的排料阀是关闭的 - 紧急关断未动作
如果满足下列条件，则泵被关断： - 在电机启动后的10秒内转速计未指示已达到额定速度

排料阀
排料阀让混合物排出（靠重力排出）到过程的下一阶段。阀门是带有弹簧的螺线管 - 如果螺线管动作则送出阀打开 - 如果螺线管不动作则送出阀关闭
送出阀由一个操作员站控制（打开/关闭）。
以下条件满足排料阀可以打开： - 搅拌电机关断 - 罐液位传感器未指示“罐空” - 紧急关断未动作
如果满足下列条件，则泵被关断： - 罐液位传感器指示“罐空”

罐液位测量开关
混合罐中的开关指示罐的液位高度关用来联锁进料泵和搅拌电机

3.4 列表输入、输出和入/出

为每个要控制的设备写出物理说明后，为每个设备或任务区域画出输入和输出图。



这个图相应于要编程的逻辑块。

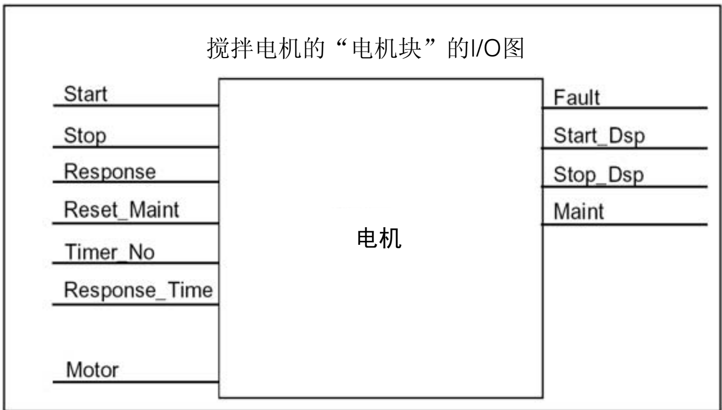
3.5 为电机生成一个 I/O 图

在我们这个工业搅拌过程的例子中使用了两个进料泵和一个搅拌电机。每个电机由它自己的功能块“motor block ”控制，而这个“motor block”对三个设备都是一样的。该块需要六个输入：两个用于启动或停止电机，一个用于复位维护显示，一个用于电机的响应信

号（电机运行/未运行），一个用于运行期间必须接收的响应信号，一个用于流量时间的定时器的号码。

逻辑块还需要4个输出：两个指示电机的操作状态，一个指示故障，一个指示电机应维护了。

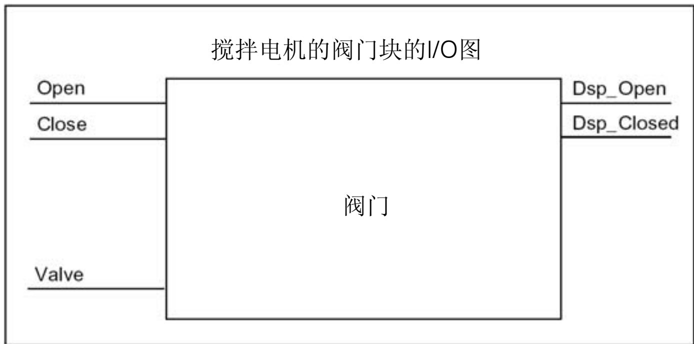
还需要一个入/出参数启动电机。它被用作控制电机但同时也在“电机块”的程序中被编辑并修改。



3.6 为阀门创建一个 I/O 图

每个阀由它自己的功能块“valve block”控制，该块对所有的阀都是一样的。该逻辑块有两个输入：一个用来打开阀，一个用来关闭阀。它还有两个输出：一个用于指示阀是打开的，另一个用于指示阀是关闭的。

该块有一个入/出参数用于启动该阀。它被用作控制阀门但同时也在“valve block”的程序中被编辑和修改。



3.7 建立安全要求

根据法定的要求及公共健康和安全政策，决定为确保过程安全还需要哪些附加组件。在你的描述中还应包括那些安全组件对你的过程区域的影响。

定义安全要求

确定哪些设备需要硬件接线电路以达到安全要求。通过定义，这些安全电路的操作独立于可编程控制器之外（虽然安全电路通常提供一个I/O接口以便与用户程序相配合）。通常你要组态一个矩阵来连接每一个执行器，这些执行器都有它自己的紧急断开范围。这个矩阵是安全电路的电路图的基础。

要设计安全机制可按如下进行：

- 决定每个自动化任务之间逻辑的和机械的/电的互锁。
- 设计电路使得属于过程的设备可以在紧急情况下手动操作。
- 为过程的安全操作建立更进一步的安全要求。

建立安全电路

在工业搅拌过程示例中使用了以下逻辑作为它的安全电路：

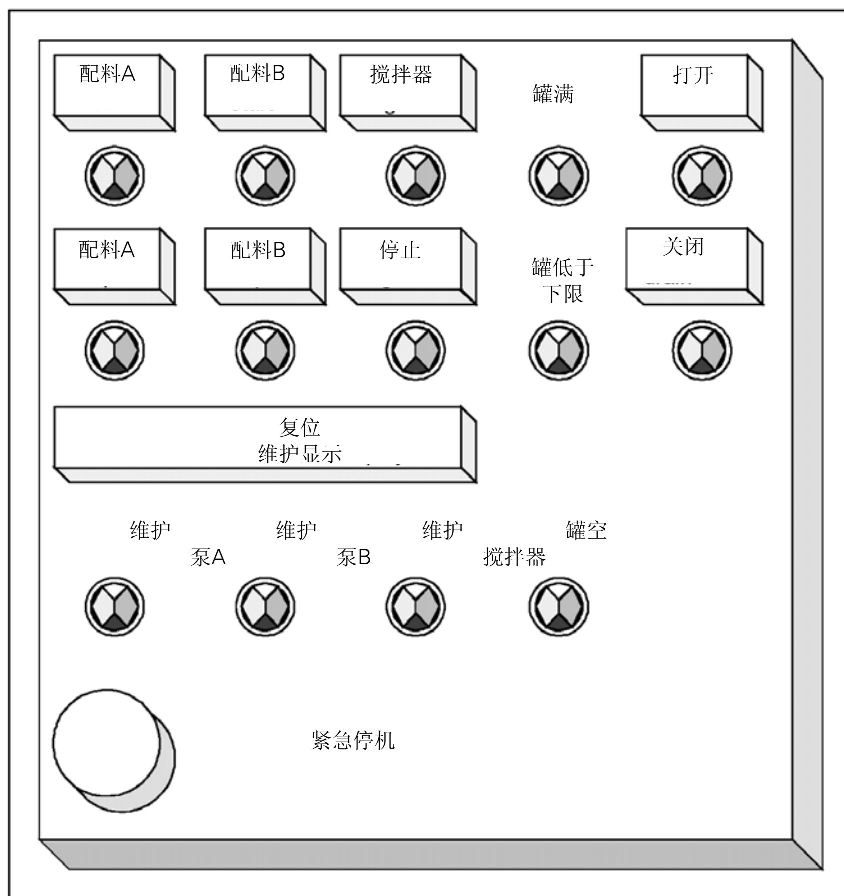
- 一个紧急断开开关可独立于可编程控制器（PLC）之外关掉以下设备：
  - 配料A的进料泵
  - 配料B的进料泵
  - 搅拌电机
  - 阀门
- 位于操作员站的紧急断开开关。
- 一个用于指示紧急断开开关状态的控制器的输入。

3.8 描述所需要的操作员显示和控制

每个过程需要一个操作接口，使得操作人员能够对过程进行干预。设计技术规范的部分包括操作员控制站的设计。

定义操作员控制站

在我们示例中所描述的工业搅拌过程，每个设备都可以由操作员控制站上的按钮来启动或停止。这个操作员控制站包括用以指示操作状态的指示灯（见下图）。



操作站上还包括指示设备在经过一定次数的启动后需要维护的指示灯以及可以使过程立即停止的紧急断开开关。操作站上还有用来复位三个电机的维护显示灯的按钮。用这个按钮可以关断用于指示电机应进行维护的维护指示灯并将相应的计数器清零。

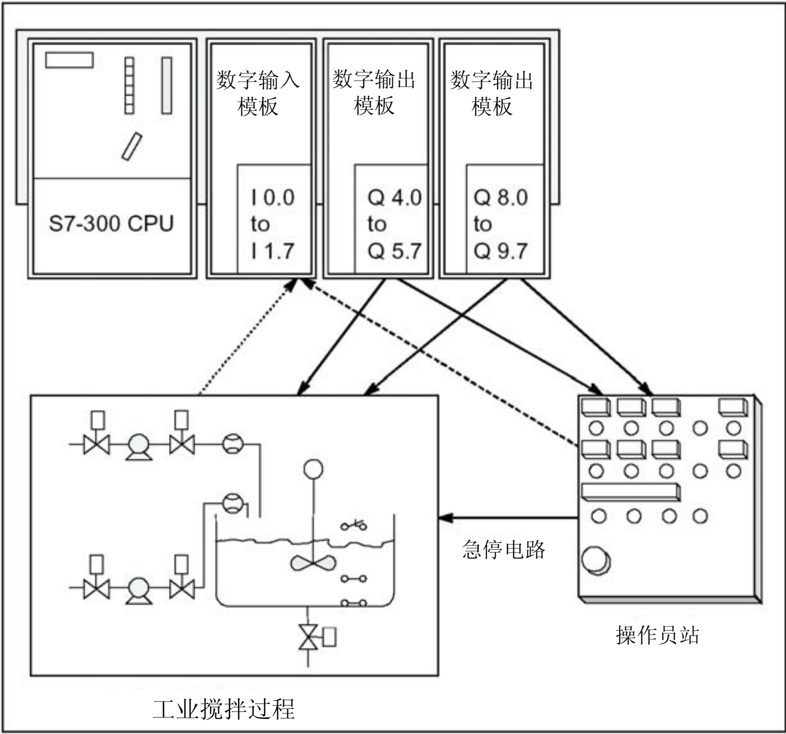
3.9 生成一个组态图

在制作了设计要求的文档后，还必须决定项目所需的控制设备的类型。

通过决定使用什么样的模板也就指定了可编程控制器的结构。生成一个组态图指定以下方面：

- CPU类型
- I/O模板的类型及数量
- 物理输入和输出的组态

下图所示为工业搅拌过程的S7组态的示例。



4 设计程序结构基础

4.1 CPU 中的程序

在一个CPU中，有两种不同的程序总会被执行：

- 操作系统
- 用户程序

操作系统

每个CPU都有一个操作系统，用以组织与特定的控制任务无关的CPU的功能和顺序。操作系统的任务包括以下各项：

- 处理暖起动和热起动
- 刷新输入的过程映象表和输出的过程映象表
- 调用用户程序
- 检测中断并调用中断OB
- 检测并处理错误
- 管理存储区域
- 与编程器和其它通讯伙伴之间的通讯。

如果修改了操作系统的参数（操作系统的缺省设置），则会影响CPU在某些区域的操作。

用户程序

你必须自己生成用户程序并下载到CPU。这个程序中包含处理你的特定的自动化任务所需要的所有功能。用户程序的任务包括以下各项：

- 指定在CPU上暖起动和热起动的条件（例如，带有某个特定值的初始化信号）
- 处理过程数据（例如，二进制信号的逻辑组合、读入并处理模拟信号、为输出指定二进制信号、输出模拟值）
- 指定对中断的响应
- 处理程序的正常运行中的干扰

4.2 用户程序中的块

STEP 7编程软件允许结构化你的用户程序，也就是说可以将程序分解为单个的、自成体系

的程序部分。这样做有以下优势：

- 大规模的程序更容易理解
- 可以对单个的程序部分进行标准化
- 程序组织简化
- 程序修改更容易
- 由于可以分别测试各个部分，查错更为简单
- 系统的调试更容易

工业搅拌过程的示例说明了将一个自动化过程分解为单个的任务的优越性。结构化的用户程序的各个部分相应于这些单个的任务，就是大家所知的程序块。

块类型

在S7用户程序中有几种不同类型的块可以使用：

块	功能的简要描述	参考
组织块（OB）	OB决定用户程序的结构	组织块和程序结构
系统功能块（SFB） 系统功能（SFC）	SFB和SFC集成在S7 CPU中可以让你访问一些重要的系统功能	系统功能块（SFB）和 系统功能（SFC）
功能块（FB）	FB是带有“存储区域”的块，你可以自己编程这个存储区域	功能块（FB）
功能（FC）	FC中包含经常使用的功能的例行程序	功能（FC）
背景数据块 （背景DB）	当一个FB/SFB被调用时，背景DB与该块相关联，它们可在编译过程中自动生成	背景数据块
数据块（DB）	DB是用于存储用户数据的数据区域，除了指定给一个功能块的数据，还可以定义可以被任何块使用的共享数据	共享数据块（DB）

OB、FB、SFB、FC和SFC都包含部分程序，因此也称作逻辑块。每种块类型所允许的块的数量以及块的长度视CPU而定。

4.2.1 组织块和程序结构

组织块（OB）是操作系统和用户程序之间的接口。它们由操作系统调用并控制循环和中断驱动的程序的执行以及可编程控制器如何启动。它们还处理对错误的响应。通过编程组织块，你可以指定CPU的动作。

组织块的优先级

组织块决定各个程序部分执行的顺序。一个OB的执行可以被另一个OB的调用而中断。哪个OB可以中断其它OB，由它的优先级决定。高优先级的OB可以中断低优先级的OB。背景OB的优先级最低。

中断的类型和优先级

导致OB被调用的事件就是所知的中断。下表显示了STEP 7中的中断类型以及分配给它们的组织块的优先级。并非所有列出的组织块和它们的优先级适用于所有的S7 CPU（见《S7-300可编程控制器、硬件和安装手册》以及《S7-400、M7-400可编程控制器模板技术规范参考手册》）。

中断类型	组织块	优先级（缺省）	参见
主程序循环	OB1	1	用于循环程序处理的组织块（OB1）
日时钟中断	OB10 至 OB17	2	日时钟中断组织块（OB10-OB17）
时间延迟中断	OB20 OB21 OB22 OB23	3 4 5 6	时间延迟中断组织块（OB20-OB23）
循环中断	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	7 8 9 10 11 12 13 14 15	循环中断组织块（OB30-OB38）
硬件中断	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	16 17 18 19 20 21 22 23	硬件中断组织块（OB40-OB47）
DPV1中断	OB55 OB56 OB57	2 2 2	编程DPV1设备
多处理器中断	OB60多处理器	25	多处理器-多CPU的同步操作
同步周期中断	OB61 OB62 OB63 OB64	25	PROFIBUS-DP的等时同步处理
冗余错误	OB70 I/O冗余错误（只适用于H系统） OB72 CPU冗余错误（只适用于H系统）	25 28	故障处理级织块（OB70-OB87/ OB121-OB122）

中断类型	组织块	优先级（缺省）	参见
异步的故障	OB80时间错误 OB81电源故障 OB82诊断中断 OB83插入/移走模板中断 OB84 CPU硬件故障 OB85 程序循环错误 OB86 机架故障 OB87 通讯错误	25 （或者如果异步错误OB存在于启动程序中则为28）	故障处理组织块 块（OB70-OB87/OB121-OB122）
背景循环	OB90	29[1)]	背景组织块（OB90）
启动	OB100暖启动 OB101热启动 OB102冷启动	27 27 27	启动组织块 （OB100/OB101/OB102）
同步错误	OB121编程错误 OB122访问错误	引起错误的OB的优先级	故障处理组织块 （OB70-OB87/OB121-OB122）
1）优先级别29相应于优先级0.29。背景循环的优先级低于自由循环。			

改变优先级

用STEP 7可以对中断进行参数赋值，比如在参数块中取消选定中断OB或优先级：日时钟中断，时间延迟中断，循环中断和硬件中断。

S7-300 CPU中组织块的优先级是固定的。

对S7-400 CPU（以及CPU318），你可以用STEP 7修改以下组织块的优先级：

- OB10至OB47
- OB70至OB72（只适用于H CPU）以及OB81至OB87在RUN模式下

下列为允许的优先级：

- 优先级2至23相应于OB10至OB47
- 优先级22或28相应于OB70至OB72
- 优先级24或26相应于OB81至OB87。对于CPU，在2001年中期（硬件版本V3.0）这些优先级做了扩展：优先级2到26除了可以分配给OB86/OB87以外，还可以分配给OB81～OB84。

可以将同一个优先级分配给几个OB。具有相同优先级的OB的处理顺序是它们的启动事件出现的顺序。

由同步错误启动的故障OB被处理的优先级与错误出现时正在执行的块的优先级一样。

局域数据

生成逻辑块（OB、FC、FB）时可以声明临时局域数据。CPU上的局域数据区域按优先级划分。

在S7-400上，可以使用STEP 7在“优先级”参数块中改变每个优先级的局域数据的数量。

OB的启动信息

每个组织块都有20个字节局域数据的启动信息，这些信息是一个OB被启动时由操作系统提供的。这些启动信息指定了OB的启动事件，OB启动的日期和时间，出现的错误及诊断事件。

例如，OB40是一个硬件中断OB，在它的启动信息中包含产生中断的模板地址。

取消选定中断OB

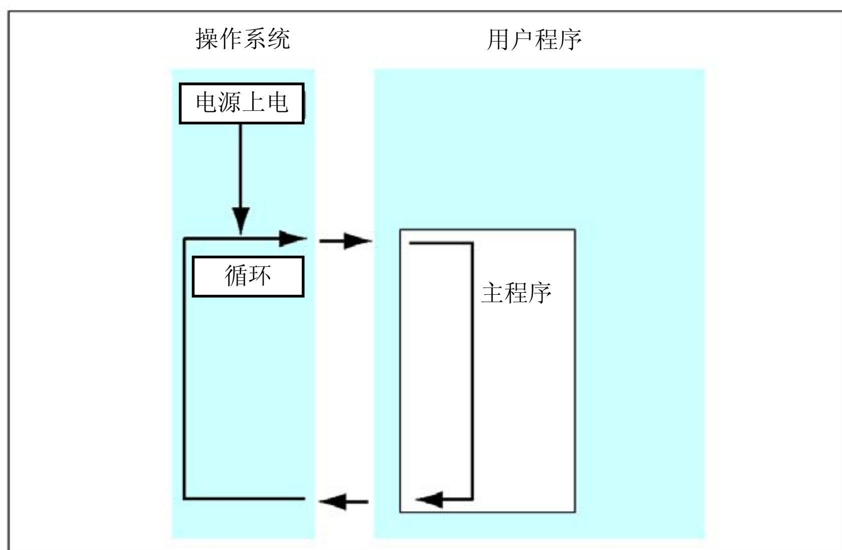
如果将优先级赋值为0，或分配少于20个字节的局域数据给某个优先级，则相应的中断OB就被取消选定。对于取消选定的中断OB的处理有以下限定：

- 在RUN模式下，它们可以被拷贝或连接到你的用户程序。
- 在STOP模式下，它们可以被拷贝或连接到你的用户程序，但是当CPU进行暖启动时，它们停止这个启动并在诊断缓存区中存入一个输入项。

通过取消选定你不需要的中断OB，可以增加可供使用的局域数据量，这可以被用来在其它优先级中节省临时数据。

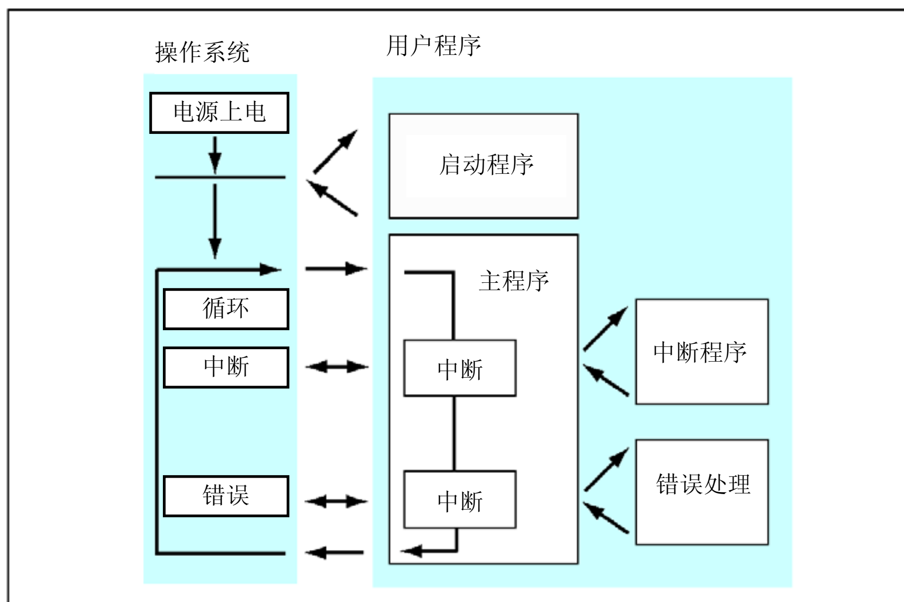
循环程序处理

循环程序处理是可编程控制器上程序执行的“普通”类型，这意味着操作系统在程序环（循环）中运行，并且在主程序的每一个环中调用一次组织块OB1。OB1中的用户程序因此也被循环执行。



事件驱动的程序处理

循环程序处理可以被某些事件中断。如果一个事件出现，当前正在执行的块在语句边界被中断，并且另一个被分配给特定事件的组织块被调用。一旦该组织块执行结束，循环程序将从断点处继续执行。

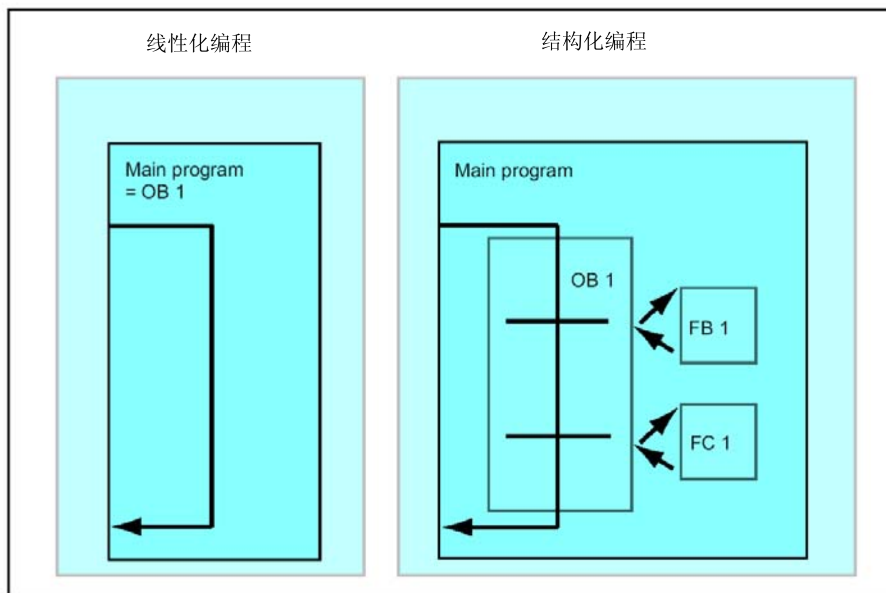


这意味着部分用户程序可以不必循环处理，只在需要时处理。用户程序可以分割为“子程序”，分布在不同的组织块中。如果用户程序是对一个重要信号的响应，这个信号出现的次数相对较少（例如，用于测量罐中液位的一个限位传感器报警达到了最大上限），当这个信号出现时，要处理的子程序就可以放在一个事件驱动处理的OB中。

线性编程与结构化编程

可以将整个用户程序写在OB1中（线性化编程）。只有在为S7-300编写简单程序并且需要较少存储区域时，才建议使用这种方法。

将复杂的自动化任务分解为能够反映过程的工艺、功能或可以反复使用的小任务时，控制会更加容易。这些任务由相应的程序部分表示，即为所知的块（结构化编程）。

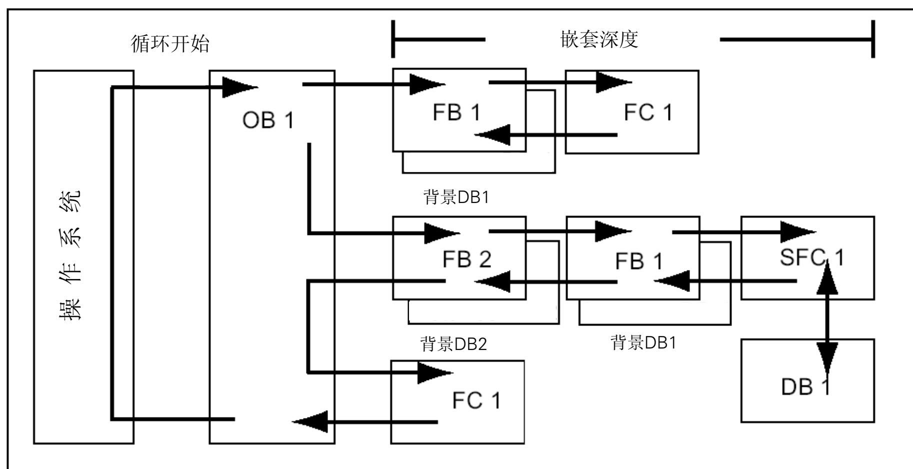


4.2.2 用户程序中调用的分层结构

为使用户程序工作，组成用户程序的块必须被调用。使用专门的STEP 7指令可以实现块调用，块调用的指令只能在逻辑块中编写和启动。

顺序和嵌套深度

块调用的顺序和嵌套深度即是所谓的调用分层结构。可以嵌套调用的块的数量（嵌套深度）依据特定的CPU而定。下图所示为一个循环周期内块调用的顺序和嵌套深度。



创建块的一套顺序：

- 创建块应从上到下，所以应从最上行的块开始。
- 每个被调用的块应已经存在，即在块的一行中应按从右向左的顺序创建它们。
- 最后创建的块是OB1。

将这些规则用于上图示例的练习，就产生以下块创建的顺序：

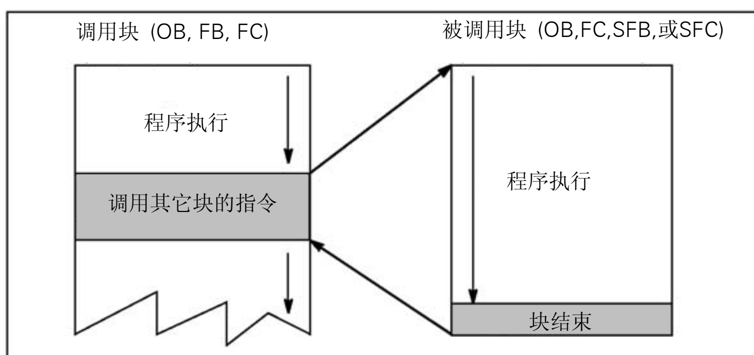
FC1>FB1+背景DB1>DB1>SFC1>FB2+背景DB2>OB1

提示

如果嵌套太深（层次太多），局域数据堆栈会溢出（又见局域数据堆栈）。

块调用

下图所示是在一个用户程序中块调用的顺序。程序调用第二个块，这个块的指令则完全被执行。一旦第二个块或者说这个被调用的块执行结束，由于执行调用指令而被中断的块的执行，将从块调用指令后面继续。



在编程一个块之前，必须指定你的程序将使用哪些数据，也就是说，必须声明块的变量。

提示

- 必须为每个块调用说明OUT参数。
- 当执行冷启动时，操作系统复位SFB3“TP”参数。如果你想在一个冷启动之后，初始化这个SFB的参数，必须通过OB100用PT=0 ms 调用该SFB的相关参数。例如，你可以通过在一个包含该SFB的块中执行一个初始化例行程序来实现这一步。

4.2.3 块类型

4.2.3.1 用于循环程序处理的组织块（OB1）

在可编程控制器上循环程序处理是程序执行的“普通”类型。操作系统循环调用OB1启动用户程序的循环执行。

循环程序执行的顺序

下表所示为循环程序处理的各个阶段：

步骤	10/98以前的CPU中的顺序	在新CPU中的顺序（10/98以后）
1.	操作系统启动循环监控时间	操作系统启动循环监控时间
2.	CPU读输入模板的输入状态并刷新输入的过程映象表	CPU从输出的过程映象表写数值到输出模板
3.	CPU处理用户程序并执行程序中所包含的指令	CPU读输入模板的输入状态并刷新输入的过程映象表
4.	CPU从输出的过程映象表写数值到输出模板	CPU处理用户程序并执行程序中所包含的指令
5.	在循环结束处，操作系统执行所有挂起的任务，例如，下载和删除块，接收和发送全局数据	在循环结束处，操作系统执行所有挂起的任务，如下载和删除块，接收和发送全局数据
6.	最后，CPU回到循环开始处并重新启动循环监控时间	最后，CPU回到循环开始处并重新启动循环监控时间

过程映象

所以在循环程序处理过程中，CPU的过程信号映象是一致的。CPU并不直接访问I/O模板上的输入（I）和输出（Q）地址区域，而是访问一个CPU内部的存储区域，该区域中包含输入和输出的映象。

编程循环程序处理

你可以通过使用STEP 7在OB1中编写你的用户程序以及在OB1调用的块中编写程序来编程循环程序处理。

启动程序无错执行一结束，循环程序处理就立即开始。

中断

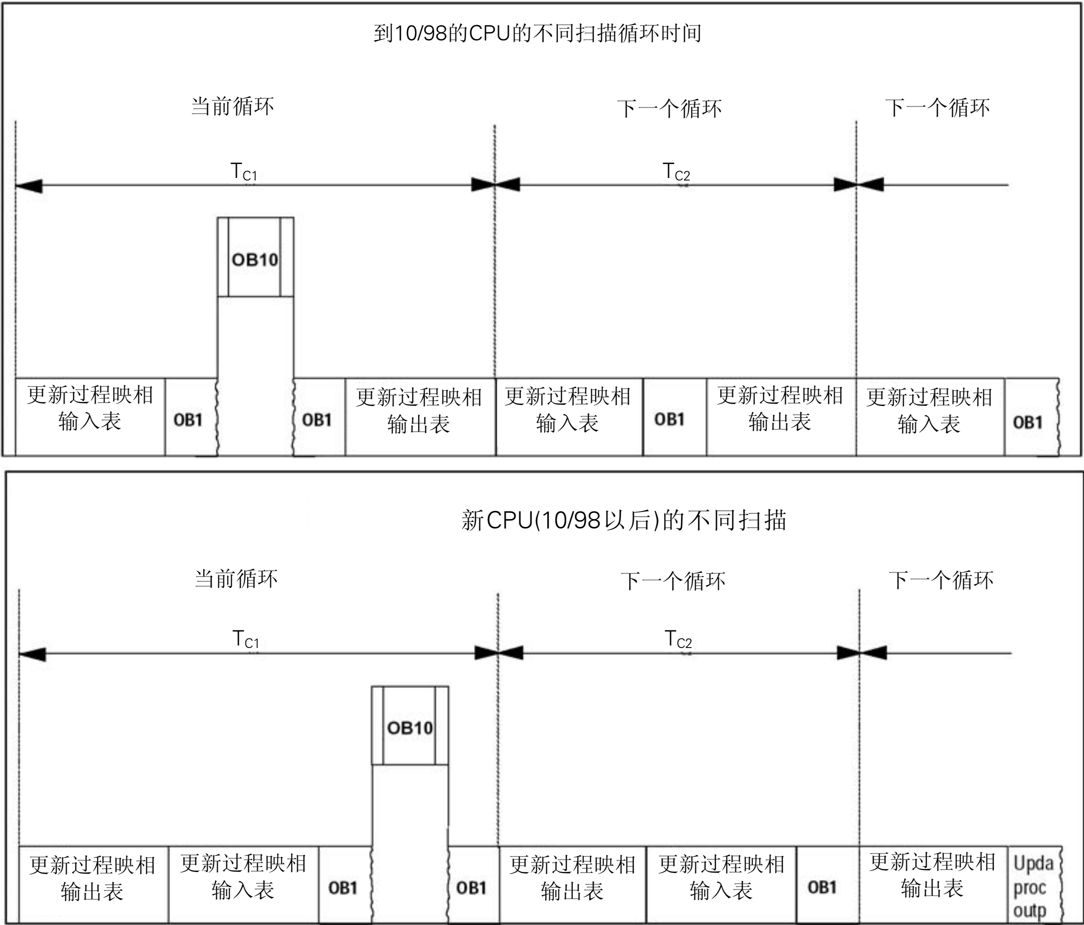
循环程序处理可以被以下事件中断：

- 一个中断
- STOP命令（模式选择开关，编程器上的菜单选项，SFC46 STP，SFB20 STOP）
- 电源掉电
- 出现故障或编程错误

扫描循环时间

扫描循环时间是操作系统运行循环程序和中断循环的所有程序部分（例如，执行其它组织块）以及系统操作（如，刷新过程映象）所需要的时间。这个时间可以被监控。

每个循环的循环扫描时间（TC）并不相同。下图所示为已有的CPU和新CPU之间不同的扫描循环时间（TC1≠TC2）：



在当前的循环中，OB1被日时钟中断所中断。

循环监视时间

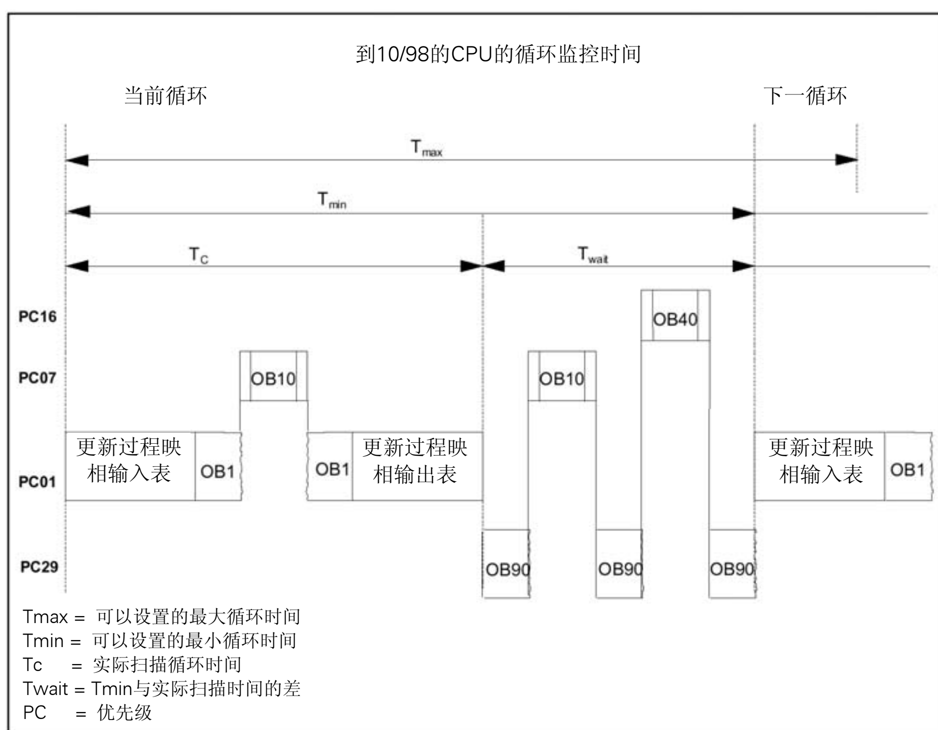
使用STEP 7，可以修改缺省的最大循环监视时间。如果这个时间溢出，CPU转为STOP模式或调用OB80，在OB80中你可以指定CPU如何响应这个故障。

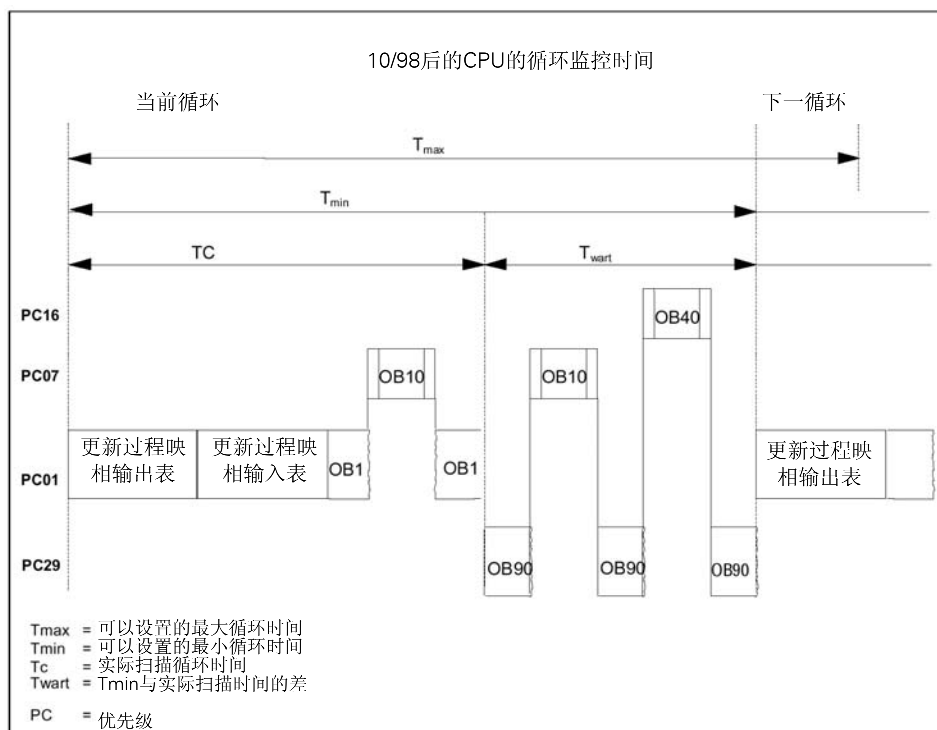
最小循环时间

使用STEP 7，可以为S7-400 CPU和CPU318设置最小循环时间。在以下情形下这个功能是有用的：

- 当OB1（主程序扫描）执行程序的启动时间间隔必须一致时，或者
- 如果循环时间太短，过程映象表的刷新没必要过于频繁。

下图所示为已有的CPU和新CPU中程序处理过程中的循环监控时间的功能。





刷新过程映象

在CPU的循环程序处理过程中，过程映象区被自动刷新。对于S7 400 CPU以及CPU 318，你也可以不刷新过程映象，如果你要：

- 直接访问I/O或者
- 用系统功能SFC26 UPDAT_PI和SFC27 UPDAT_PO在程序的不同点刷新一个或多个过程映象的输入或输出部分。

通讯负载

你可以使用CPU参数“Scan Cycle Load from Communication（扫描通讯循环负载）”，在给定框架内控制通讯过程的持续时间，因为一般通讯过程会增加扫描循环时间。例如通讯过程包括以MPI方式将数据传送至其它CPU，或使用编程设备加载的通讯功能块。

该参数很少影响编程器测试功能。当然，你可以显著增加扫描时间。在处理模式中，你可以限制测试功能的时间（只适用于S7-300）。

参数的工作原理

CPU操作系统可持续提供用户组态的整个CPU处理能力的百分比（时间片技术）用于所组态的通讯。但如果不需要处理通讯时，可用于其它处理。

设置扫描循环时间

无需其它异步事件，OB1扫描循环时间可以根据以下公式乘以一个系数进行扩展：

$$\frac{100}{100 - \text{通讯扫描循环负载（\%）}}$$

举例1（无其它异步事件）：

如果你设定通讯循环加载50%，OB1循环时间将加倍。

同时，OB1扫描循环时间还会受到异步事件的影响（例如硬件中断或循环中断）。根据统计学观点，由于扫描循环时间通讯部分的扩展，在OB1扫描循环时间内将有更多异步事件发生。这会导致OB1扫描循环的额外增加。这种增加取决于每个OB1扫描循环时间发生的事件数量以及事件处理时间。

举例2（考虑其它异步事件）：

对于一个执行时间为500ms的OB1来说，50%的通讯负载将会导致实际扫描循环时间为1000ms（倘若CPU总是有足够的通讯作业要处理）。假如先不考虑通讯负载，每100ms执行一次处理时间为20ms的循环中断，这种循环中断会增加扫描循环 5*20 ms = 100 ms。即，实际扫描循环时间将为 600 ms。但由于一个循环中断也会中断通讯，因此50%的通讯负载会增加扫描循环时间10 * 20 ms，即在这种情况下，实际扫描循环时间为1200 ms，而不是1000 ms。

提示

- 检查系统运行时“通讯循环负载”参数的改变效果。
- 在设置最小扫描循环时间时要考虑通讯负载，否则会出现时间错误。

建议

- 如果可能的话，使用缺省值。
- 只有将CPU用于通讯目的，并且用户程序没有执行周期方面的影响，才能增加该数值。
- 否则，应降低该数值。
- 设定处理模式（只适用于S7-300），限制测试功能所需时间。

4.2.3.2 功能（FC）

功能（FC）属于你自己编程的块。功能是“无存储区”的逻辑块。FC的临时变量存储在局域数据堆栈中。当FC执行结束后，这些数据就丢失了。要将这些数据永久存储，功能也可以使用共享数据块。

由于FC没有它自己的存储区，所以你必须为它指定实际参数。不能够为一个FC的局域数据分配初始值。

应用程序

一个FC包含一个程序部分，当FC被不同的逻辑块调用时，这些程序总会被执行。可为以下目的使用功能：

- 为调用块返回一个功能值（例如：数学功能）
- 要执行一个工艺功能（例如：使用位逻辑操作的单个控制功能）

将实际参数赋值给形式参数

形式参数是“实际”参数的虚名称。当功能被调用时，用实际参数替代形式参数。对于FC来说，形式参数总是必须赋给实际参数（例如：将实参“I 3.6”赋值给形参“Start”）。输入、输出和输入/输出参数被FC做为指向实参的指针，当FC调用时指向的实际参数。

FC和FB输出参数的主要区别

在功能块（FB）中，当访问参数时使用背景数据块中的实际参数的拷贝参数。当调用FB时，如果没有传送输入参数或没有写输出参数，则背景数据块（FB的存储区）中将始终保持以前的值。

功能（FC）没有存储器，与FB对比，不可以选择对FC的形参赋值。FC的参数只能通过寻址来访问（类似于交叉区域指针）。当数据区（数据块）中的的一个地址或的局部变量作为实际参数被调用时，一个复制的实际参数将被存储到调用块的临时的局部数据区，用它来传送数据。

注意：

在这种情况下，如果没有向FC的输出参数写入一个数据，则将输出一个随机值。

由于作为复制数据所保留的调用块的局部数据区没有赋值到输出参数，所有该区没有写入任何数据。因此将输出存储在该区域的随机值，因为局部数据不能自动地设置为0。

因此，请遵守下列准则：

- 如果可能，对输出参数初始化。
- 根据RLO执行置位和复位指令。当这些指令用来决定输出参数的值时，如果前一次的逻辑运算RLO=0，则不会生成数值。
- 确保写到输出参数中的数据与块中的任何程序路径无关。要特别注意跳转指令、LAD和FBD中的ENO输出指令、BEC（Block End Conditional）指令以及MCR（Master Control Relay）指令。

注意：

尽管一个FB的OUTPUT参数或FC和FB的INOUT参数不输出随机值(即使没有向参数写入数据，旧的输出值-或将输入作为输出的值将保持输出)，也应该遵守上述规则，以避免无意中对“旧”值进行处理。

4.2.3.3 功能块（FB）

功能块（FB）属于用户自己编程的块。功能块是具有“存储功能”的块。数据块作为功能块的存储器（背景数据块）被分配给FB。传递给FB的参数和静态变量都保存在背景数据块中。临时变量存在本地数据堆栈中。

当FB执行结束时，存在背景DB中的数据不会丢失。可是，当FB的执行结束时，存在本地数据堆栈中的数据将丢失。

提示

为避免使用FB时出现错误，请参照附录中的“传递参数时所允许的数据类型”。

应用程序

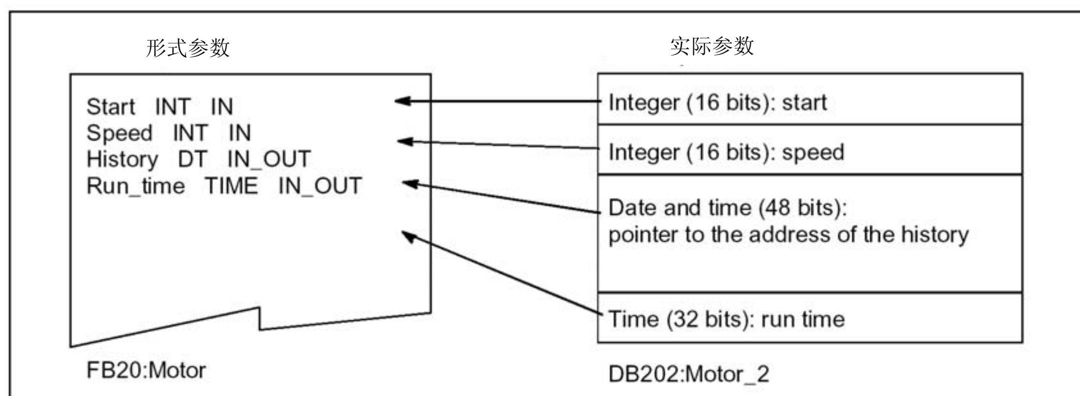
当FB被调用时，FB中所含的程序总被执行。功能块使得对于经常使用的功能、复杂功能的编程变得容易。

功能块和背景数据块

一个背景数据块被指定给每一个被调用的功能块（FB）被称为参数传递。

通过调用同一个FB的不同的背景数据块，用户可以用一个FB控制多台设备。例如，一个用于电机控制的FB，可以通过对每个不同的电机，使用不用的背景数据，来控制多台电机。每台电机的数据（例如：转速、爬升、累积运行时间等），可存在一个或多个背景数据块中。

下图所示为FB的形参用实参赋值后并存在背景DB中的情况。



数据类型 FB

如果用户程序的结构是，在一个FB中又调用了另一个已经存在的FB功能块，用户可在调用FB的变量定义表中将被调FB作为的静态数据类型为FB的变量。这将允许用户嵌套变量，并将背景数据压缩在一个背景数据块（多重背景）中。

将实际参数赋值给形参

在STEP 7中，对于FB通常不是必须将实际参数赋值给形参。可是，下列情况除外，对以下形参必须赋实参：

- 对于复杂数据类型（如：字符串（STRING），数组（ARRAY）或日期与时间（DATE__AND__TIME）的输入/输出类型参数。
- 对于所有的参数类型(例如,定时器(TIMER)、计数器(COUNTER)或指针(POINTER)。

STEP 7会按照如下方式将实际参数赋值给FB的形式参数：

- 当用户在调用语句中定义了实际参数时：FB的指令使用所提供的实参。
- 当用户在调用语句中没有定义实际参数：FB的指令使用存在背景DB中的值。

下表所示为哪些FB的变量必须赋实参。

变　量	数据类型		
	基本数据类型	复杂数据类型	参数类型
输入	无实参要求	无实参要求	有实参要求
输出	无实参要求	无实参要求	有实参要求
输入/输出	无实参要求	有实参要求	–

给形参赋初值

在FB的声明表中，用户可给形式参数赋初值。这些值将写入与FB相关的背景DB中。

如果用户在调用语句中，没有给形参赋实参，则STEP 7将使用存在背景DB中的值。这些值也可作为初值输入到FB的变量定义表中。

下表所示为哪些变量可以赋初值。由于临时数据在该块执行完后将丢失，所以用户不能给它们赋任何值。

变　量	数据类型		
	基本数据类型	复杂数据类型	参数类型
输入	允许有初值	允许有初值	–
输出	允许有初值	允许有初值	–
输入/输出	允许有初值	–	–
静态	允许有初值	允许有初值	–
临时	–	–	–

4.2.3.4　背景数据块

一个背景数据块被分配给一个功能块（FB）被称作参数传递。FB的实际参数和静态数据存在背景DB中。在FB中定义的变量，决定背景数据块的结构。“背景”意味着一次功能块调用。例如，如果在S7用户程序中某个功能块被调用了五次，则该块有五个“背景”。

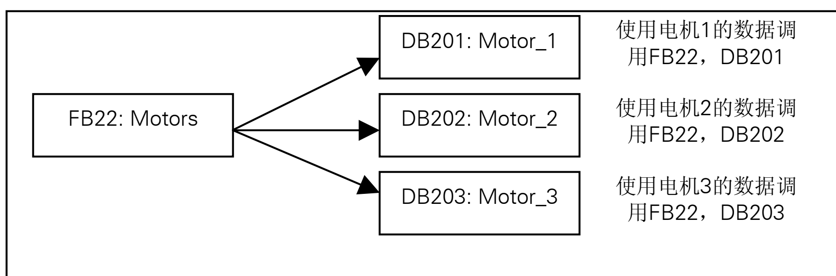
生成一个背景DB

在用户生成一个背景数据块之前，相应的FB必须已经存在。当用户生成背景数据块时，必须指定所属FB的序号。

每次不同的背景用一个背景DB

如果用户将多个背景数据块分配给某个控制电机的功能块（FB），则用户可用该FB去控制多个不同的电机。

描述电机的各项数据（例如：转速、升速时间、整个运行时间），存在不同的数据块中，当FB调用时，相应的DB决定哪个电机被控制。这样，控制多台电机只需一个功能块（参看下图）。

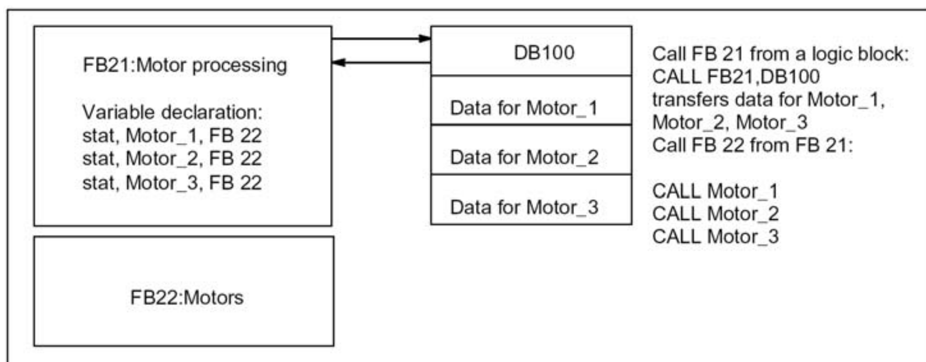


一个背景DB用于某个FB的多次背景（多重背景）

用户也可以将多个电机的背景数据同时传递到一个背景DB。为此，用户必须增加一个FB来管理电机控制器的多次调用，并且，在调用FB的定义表中用数据类型为“FB”的静态变量定义每个背景。

如果用户只用一个背景DB存放某个FB的多次背景，则节约了存储空间，并能最优地使用数据块。

在下图中，调用FB是FB21“电机控制”，变量为数据类型FB22，不同的背景用“Motor-1”、“Motor-2”和“Motor-3”标识。



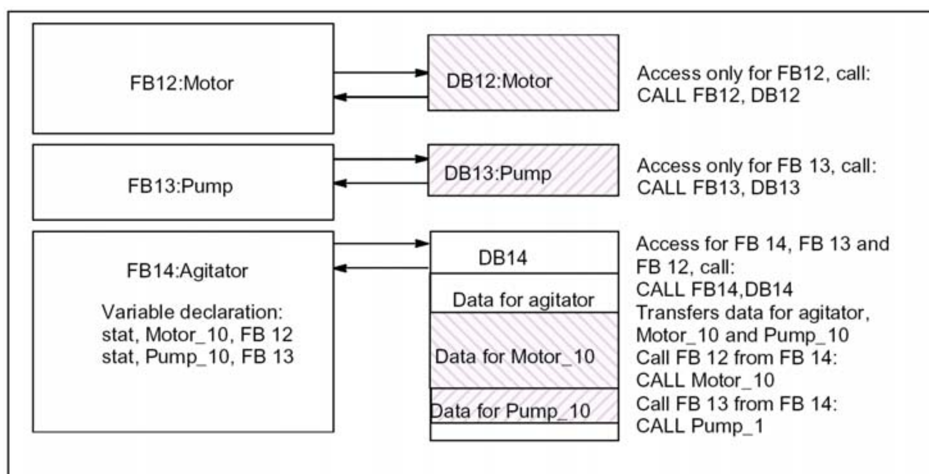
在这个例子中，FB22不需要自己的背景数据块，因为它的背景数据存在调用FB的背景数据块中。

一个背景DB用于不同FB的多次背景（多重背景）

在一个功能块中，用户可以调用其它已经存在的FB的背景。为此，用户可以将所需的背景数据赋值到调用FB的背景数据块中，这就意味着，在这种情况中，用户不需要为被调FB增加任何数据块。

为了将这些多重背景在一个背景数据块中实现，用户必须在调用功能块的变量声明表部分，为每次独立的背景定义一个以被调用的FB为数据类型的静态变量。在功能块内部的调用，则不再需要背景数据块，只需要变量的符号名。

下图示例中，赋值的背景数据存在一个共同的背景DB中。



4.2.3.5 共享数据块（DB）

与逻辑块不同，在数据块中没有STEP 7的指令。它们用于存放用户数据，换句话说，数据块中存放用户程序工作时所需的变量数据。共享数据块用于存放所有其它块都可以访问的用户数据。

DB的大小可以不同。关于所允许的最大尺寸，请参考用户所用CPU的描述。

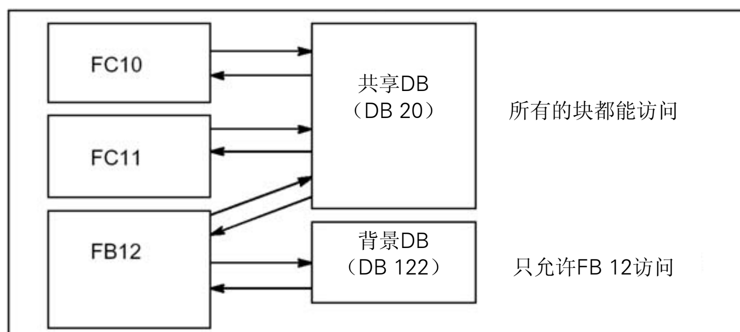
用户可以用任意方式来建立数据块的结构，以适合其不同的需求。

在用户程序中的共享数据块

如果某个逻辑块（FC，FB或OB）被调用，则它可以临时占用临时本地数据区的空间（L堆栈）。除了这个本地数据区，逻辑块还可以打开一个DB形式的存储区。与局域数据区中的数据不同，当相应的逻辑块运行结束从而DB关闭时，在DB中的数据不会被删除。

每个FB、FC或OB可从共享DB中读取数据，或将数据写入共享DB。当该DB退出时，这些数据保持在DB中。

一个共享DB和一个背景DB可同时打开。下图所示为访问数据块的不同方法。



4.2.3.6 系统功能块（SFB）和系统功能（SFC）

已经编好程序块

用户不需要每个功能都自己编程。S7 CPU为用户提供了一些已经编好程序块，这些块可在用户程序中进行调用。

在系统功能块和系统功能中的参考帮助中可找到进一步的信息。

系统功能块

系统功能块（SFB）是集成在S7 CPU中的功能块。SFB作为操作系统的一部分，不占用户程序空间。与FB相同，SFB也是“具有存储器”的块。用户也必须为SFB生成背景数据块，并将其下载到CPU中作为用户程序的一部分。

S7 CPU提供下列SFB：

- 通过组态连接用于通讯目的。
- 集成的特殊功能（例如：CPU 312 IFM 和CPU 314I FM上的 SFB 29“HS_COUNT”）。

系统功能

系统功能是集成在S7 CPU中预先编好的功能。可在用户程序中调用。SFC属于操作系统的一部分，而不算做用户程序的一部分。与FC相同，SFC是“不具有存储器”的块。

S7 CPU提供以下功能的SFC：

- 复制及块功能
- 检查程序
- 处理时钟和运行计时
- 传递数据集
- 在多CPU状态中将事件从一个CPU传到所有其它的CPU中
- 处理日期时间中断和延时中断
- 处理同步错误、中断错误和异步错误
- 有关静态和动态系统数据的信息，例如：诊断

- 过程映象刷新和位域处理
- 寻址模板
- 分布式I/O
- 全局数据通讯
- 无组态连接的通讯
- 生成块相关信息

其它信息

有关SFB和SFC更详细的信息，请参考《S7-300和S7-400系统软件，系统和标准功能》参考手册，《S7-300可编程序控制器，硬件和安装手册》以及《S7-400，M7-400可编程序控制器模板特性参考手册》中关于SFB和SFC的解答。

4.2.4 用于中断程序处理的组织块

4.2.4.1 日时钟中断组织块（OB10-OB17）

S7 CPU提供日期时间中断OB，这些OB可以在特定的日期或以一定的时间间隔执行。

日期时间中断可按如下方式触发：

- 在某特定时间执行一次。
- 从特定的时间开始并按中断应重复的间隔（例如：每分钟、每小时、每天）周期地执行。

日期时间中断规则

日期时间中断只有当该中断设置了参数，并且在相应的组织块存在与用户程序中时才能被执行。如果与上述情况不符，则操作系统会在诊断缓存器中输入一个错误信息，并执行异步错误处理（OB80，请参看错误处理组织块（OB70到OB87/OB121到OB122））。

周期的日期时间中断必须对应一个实际日期。从一月三十一日开始每月重复OB10是不可能的。在这种情况下，该OB将只在有31天的那个月起动。

日期时间中断在PLC起动时（暖起动或热起动）激活，而且，只能在PLC起动过程结束之后，才能执行。

没有选中的日期时间中断不能起动。当CPU识别到调用该OB产生编程错误时，将进入到停机状态。

暖起动之后，日期时间中断必须重新设置（例如，在PLC起动程序中用SFC30 ACT_TINT）。

起动日期时间中断

为了让CPU起动日期时间中断，用户必须首先设置日期时间中断，然后再激活它。起动该中断有以下三种方法：

- 通过STEP 7中设置相应的参数（“日期时间中断”参数块），实现日期时间中断的自动起动。
- 在用户程序中用SFC28 SET_TINT和SFC 30 ACT_TINT，设置并激活日期时间中断。
- 用STEP 7的参数设置日期时间中断，在用户程序中用SFC 30 ACT_TINT激活日期时间中断。

查询日期时间中断

要想查询设置了哪些日期时间中断，以及这些中断什么时间发生，用户可选用下列方法之一：

- 调用SFC 31 QRY_TINT。
- 打开系统状态表的“中断状态”表。

终止日期时间中断

用SFC 29 CAN_TINT,用户可以取消那些还没有执行的日期时间中断。用SFC 28 SET_TINT可以重新设置那些被终止的日期时间中断，并可用SFC30 ACT_TINT重新激活它们。

日期时间中断OB

所有八个日期时间中断OB具有相同的预置优先级（2），因此，它们的优先级是按起动事件发生的顺序进行处理的。当然，用户可以通过选择适当的参数来改变优先级。

改变设置时间

用户可以用如下方法改变中断的日期时间设置：

- 用主时钟同步主站和从站的时间。
- 在用户程序中可以调用SFC0 SET_CLK设置一个新的时间。

改变时间的响应

下表所示为日期时间中断在时间已经改变之后，是如何反应的。

如果…	则…
如果时间向前调整，使有一个或多个日期时间中断已经被跳过	起动OB80，那些跳过的日期时间中断将记录到OB80的起动信息中
用户没有终止在OB80中跳过的日期时间中断	跳过的那些日期时间中断也将不再执行
用户在OB80中没有终止被跳过的日期时间中断	第一个跳过的日期时间被中断执行。其它的跳过的日期时间中断将忽略
将时间设置向后调整，日期时间中断的起动事件会再次发生	在S7-300 CPU中将重复执行日期时间中断，但在S7-400CPU中以及CPU318中不再重复执行日期时间中断

4.2.4.2 时间延迟中断组织块（OB20-OB23）

S7 CPU提供延时OB用于在用户程序中编写延时执行的程序。

延时中断规则

延时中断只有当相应的组织块在CPU程序中存在时才能执行。否则，操作系统会在诊断缓存器中输入一个错误信息，并执行异步错误处理（OB80，请参看错误处理组织块（OB70到OB87/OB121到OB122））。

在参数设置时，如果没选延时中断OB，则该OB不能起动。CPU认为这是一个编程错误，将进入到停机状态。

当在SFC 32 SRT_DINT中设定的延时时间达到时，触发延时中断。

起动延时中断

为了起动延时中断，用户必须在SFC 32中设定延时时间，当相应的时间延迟达到时，该中断OB被调用。请参考《S7-300可编程序控制器，硬件及安装手册》和《S7-400，M7-400可编程序控制器模板规范参考手册》中有关最大允许的延迟时间长度。

延时中断OB的优先级

延时中断OB优先级的缺省设置值为3至6级。用户可以设置参数改变优先级。

4.2.4.3 循环中断组织块（OB30-OB38）

S7 CPU提供循环中断OB，可用于按一定时间间隔中断循环程序的执行。

循环中断按间隔触发。间隔的时间是从STOP状态到RUN时开始计算。

循环中断的规则

当用户定义时间间隔时，必须确保在两次循环中断之间的时间间隔足够处理循环中断自己的程序。

如果用户在设置参数时，没有选循环中断OB，它们将不再起动。CPU认为这是一个编程错误，将进入到停机状态。

起动循环中断

为了起动循环中断、用户必须在STEP 7中的循环中断参数块里定义时间间隔。时间间隔必须是1ms基本时钟率的整数倍。

时间间隔=n×基本时钟率1ms

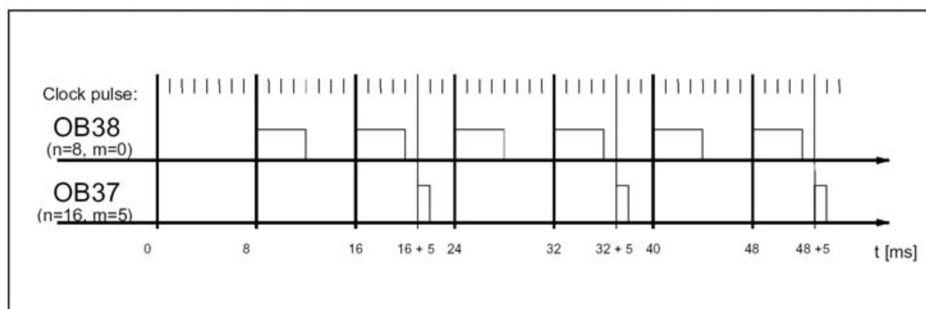
一共有九个循环中断OB，每个OB都有其缺省的时间间隔（请参看下表）。为相应的循环中断OB装载后，其对应的缺省时间间隔变为有效。然而，用户可以设置参数改变缺省值。请参阅《S7-300可编程序控制器，硬件和安装手册》及《S7-400，M7-400可编程序控制器模板规范参考手册》中有关上限的内容。

循环中断中的相位偏移

为避免不同的循环中断OB同时请求中断而可能造成的时间错误（循环时间超过），用户可以定义一个相位偏移。相位偏移确保当循环中断的时间间隔达到时，再做一定的延时。

相移=m×基本时钟率（0≤m<n）

下图所示为与无相移循环中断（OB38）相比，相移循环中断OB（OB37）的执行情况。



循环中断OB的优先级

下表所示为循环中断OB缺省的时间间隔和优先级。用户可以设置参数来改变时间间隔和优先级。

循环中断	时间间隔（ms）	优先级
OB30	5000	7
OB31	2000	8
OB32	1000	9
OB33	500	10
OB34	200	11
OB35	100	12
OB36	50	13
OB37	20	14
OB38	10	15

4.2.4.4 硬件中断组织块（OB40-OB47）

S7 CPU提供有硬件中断OB，用于对模板（例如，信号模板（SM），通讯处理器（CP），功能模板（FM））上的信号变化进行响应。通过STEP 7，用户可以决定从可组态的数字量或模拟量模板来的哪些信号可以起动OB。对于CP和FM，可能使用在对话框中设置相应的参数来起动OB。

当一个信号模板有中断能力，使能该属性后，当CPU接收到来自该模板的中断请求信号或者来自FM模板的中断信号时，将触发硬件中断。

硬件中断的规则

硬件中断只有当CPU的程序中存在相应的组织块时，才能执行。否则会在诊断缓存器中输入一个错误信息，并执行异步错误处理（OB80，请参看错误处理组织块（OB70到OB87/OB121到OB122））。

如果用户在参数设置中没有选中硬件中断OB，则它们不能起动。CPU认为这是一个编程错误，将进入到停机状态。

具有硬件中断能力的信号模板的参数设置

具有硬件中断能力信号模板的每个通道都可以触发一个硬件中断。为此，用户通过STEP 7必须给具有硬件中断能力的信号模板设置如下参数集：

- 用什么触发一个硬件中断
- 哪种硬件中断OB将被执行缺省设置（OB40用于执行所有的硬件中断）

用户通过STEP 7，可以使用功能块激活硬件中断的生成。在这些功能模块的参数设置对话框中，设置保持的参数。

硬件中断OB的优先级

硬件中断OB的缺省优先级为16到23。用户可以设置参数改变优先级。

4.2.4.5 启动组织块（OB100/OB101/OB102）

起动类型

起动特性有三种不同的类型：

- 热起动（在S7-300和S7-400H中没有）
- 暖起动
- 冷起动

下表所示为：对应各种起动类型，操作系统调用不同的OB。

起动类型	相关OB
热 起 动	OB101
暖 起 动	OB100
冷 起 动	OB102

起动OB的起动事件

当下列事件发生后，CPU执行起动功能：

- 电源上电后
- 用户将CPU的状态选择开关从“STOP”拔到“RUN/RUN-P”后
- 从通讯功能来的请求后

- 多CPU方式同步之后
- H系统中连接后（只适用于备用CPU上）

根据起动事件、所使用的CPU及其设置参数，调用相应的起动OB（OB100、OB101或OB102）。

启动程序

用户可以通过在暖起动的组织块OB100、热起动的组织块OB101或冷起动的组织块OB102中编写程序，来设定其CPU起动的起动条件（运行时的初值，I/O模板的起动值）。

起动程序没有长度限制，也没有时间限制，因为循环监视还没激活。在起动程序中，不能执行周期或中断程序。在起动过程中，所有数字量输出的信号状态都为“0”。

手动起动后的起动类型

S7-300 CPU只允许手动暖起动，只有CPU 318-2可以冷起动。

对于某些S7-400 CPU，如果用户通过STEP 7的参数设置允许手动起动，则用户可以使用状态选择开关和起动类型开关（CRST/WRST），进行手动起动。手动暖起动可以不用设定参数。

自动起动后的起动类型

对于S7-300 CPU，电源上电后，只允许暖起动。

对于S7-400 CPU，用户可定义电源上电后，是自动暖起动还是自动热起动。

模板存在/类型监视

当用户进行组态表的参数设置时，可以决定是否需要在组态表中检查模板是否存在，以及模板类型与起动前是否匹配。

如果模板检查激活，CPU发现组态表与实际组态不相符时，CPU将不起动。

监视时间

为确保可编程序控制器起动时没有错误，用户可以选择以下监视时间：

- 模板传递参数的最大允许时间
- 上电后，模板准备好用于操作的信号所允许的最大时间
- 对于S7-400 CPU，热起动期间中断所允许的最大时间

一旦监视时间达到，CPU将进入到停机状态或只能做暖起动。

4.2.4.6 背景组织块（OB90）

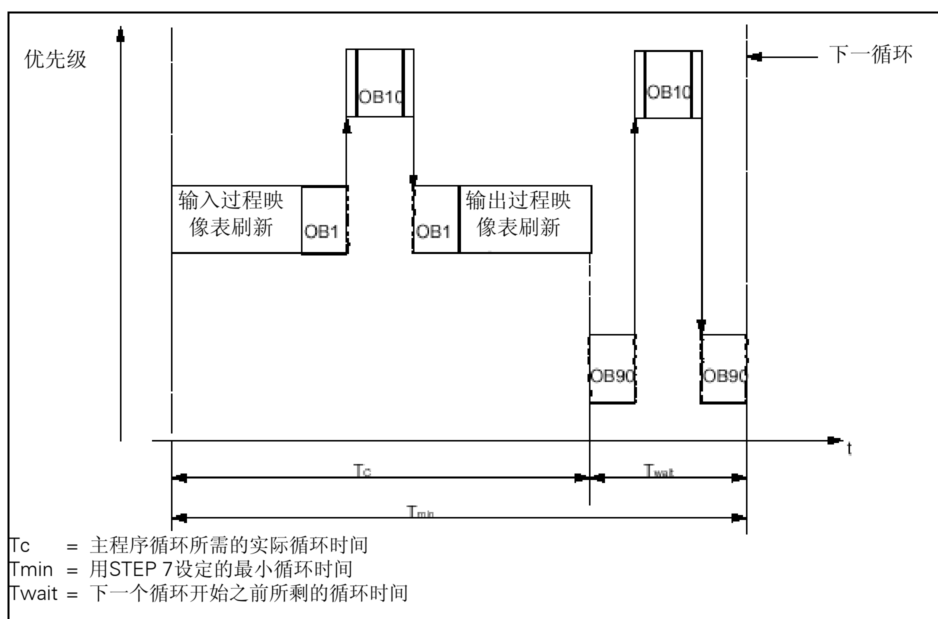
如果用户用STEP 7定义最小的扫描循环时间，且该时间比实际的扫描循环时间长，则CPU在循环程序结束时，还有处理时间。该时间用于执行背景OB。如果用户的CPU中没有OB90，

则CPU等待，直到定义的最小扫描循环时间达到为止。因此，对于那些对运行时间要求不高的过程，用户可以调用OB90，而避免等待时间。

背景OB的优先级

背景OB的优先级为29，对应的优先权0.29。因此，该OB的优先级最低。其优先级不能通过参数设置进行修改。

下图所示为，在CPU中处理的背景循环、主程序循环和OB10（CPU为10/98）。



OB90的编程

由于OB90的运行时间不受CPU操作系统的监视，因此，用户可以在OB90中编写程序的长度不受限制。为确保在背景程序中的数据具有一致性，在编程时注意以下问题：

- OB90的复位事件（参看《S7-300和S7-400的系统软件、系统和标准功能》参考手册）
- 过程映象的刷新与OB90不同步

4.2.4.7 故障处理组织块（OB70-OB87/OB121-OB122）

错误类型

可被S7 CPU检测到，并且，用户可以通过组织块对其进行响应的错误，可分为两个基本类型：

- 同步错误：这些错误作为用户程序的某一特定部分。该类错误发生在执行某特定指令过程中。如果没有装载相应的同步错误OB，当错误发生时，CPU进入到STOP。

- 异步错误：这些错误不会出现在用户程序的执行过程中。该类错误是优先级错误、可编程序控制器故障（例如：模板损坏）或冗余错误。如果没有相应异步错误OB，当错误发生时，CPU进入到“STOP”状态。（例外：OB70，OB72，OB81，OB87）。

下表所示为可能出现的错误类型，按错误处理OB进行分类。

异步错误/冗余错误	同步错误
OB70 I/O冗余错误（只适用于H CPU）	OB121编程错误（例如没有装入DB）
OB72 CPU冗余错误（只适用于H CPU，例如，一个CPU发生故障）	OB122 I/O存取错误（例如，访问一个并不存在的信号模块）
OB80计时错误（例如，超过扫描循环时间）	
OB81 电源故障（例如，电池故障）	
OB82诊断中断（例如，输入模块短路）	
OB83拆除/插入中断（例如，拆除一个输入模块）	
OB84 CPU 硬件故障（ MPI网络接口故障）	
OB85 优先级错误（例如没有装入OB）	
OB86 机架故障	
OB87通讯错误（例如，一个不正确的全局数据通讯消息帧ID）	

同步错误OB

同步错误发生在执行某一特定指令的过程中。当这些错误出现时，操作系统在I堆栈做一个输入记录，并调用同步错误OB。

该类型错误OB的调用，是由于在程序中执行了一个同步错误的结果，其优先级与检测到错误的块一致。因此，OB121和OB122可以访问中断发生时的累加器与其它寄存器中的内容。用户可利用这些值，对错误条件进行响应，然后，返回处理用户程序（例如，如果访问某个模拟输入模板时，有个访问错误，则用户可以在OB122中用SFC44 RPL_VAL定义一个替代值）。可是，该错误OB的本地数据，在L堆栈中的这个优先级中，额外占用一个空间。对于S7-400 CPU，一个同步错误OB可以起动另一个同步错误OB。对S7-300 CPU则没有这个功能。

异步错误OB

如果CPU的操作系统检测到一个异步错误时，将起动相应的错误OB（OB70到OB73和OB80到OB87）。该类异步错误OB一般具有最高等级的优先级，即便其他类型的异步错误OB的优先级与之相同，它们也不能被其它OB中断。如果同时有多个相同优先级的异步错误OB出现，它们将按其出现的顺序进行处理。

屏蔽起动事件

利用系统功能（SFC），用户可以屏蔽、延迟或禁止各种OB的起动事件。有关这些SFC和组织块的详细信息，请参考《S7-300和S7-400系统软件，系统和标准功能》参考手册。

错误OB的类型	SFC	SFC的功能
同步错误OB	SFC 36 MSK_FLT	屏蔽某个同步错误。屏蔽的错误不再起动错误OB，也不会触发编程响应
	SFC 37 DMSK_FLT	取消同步错误屏蔽
异步错误OB	SFC 39 DIS_IRT	禁止所有的中断和异步错误。被禁止的错误，在CPU的任何循环中不再起动错误OB
		也不触发编程响应
	SFC40 EN_IRT	便能中断和异步错误
	SFC 41 DIS_AIRT	延迟优先级高的中断，以及异步错误，确保该类OB执行结束
	SFC 42 EN_AIRT	使能优先级高的中断和异步错误

提示

如果用户希望忽略中断，更有效的方法是禁止它们，而不是下载一个空的OB（内容只有BE）。

5 启动和操作

5.1 启动 STEP 7



启动Windows以后，你就会发现一个SIMATIC Manager（SIMATIC管理器）的图标，这个图标就是启动STEP 7的接口。

快速启动STEP 7的方法：将光标选中SIMATIC Manager这个图标，并双击。打开SIMATIC管理器窗口。从这里，你可以访问你所安装的标准软件包和可选软件包的所有功能。

启动STEP 7的另一方式：在操作系统的任务栏中选中“Start”键，而后进入“Simatic”。

注意

你可以从STEP 7窗口的用户引导Windows打开程序在线提示，得到更多的有关标准窗口的操作及选项的信息。

SIMATIC 管理器

SIMATIC管理器用于基本的组态和编程。SIMATIC管理器具有下列功能：

- 建立项目
- 硬件组态及参数设定
- 组态硬件网络
- 编写程序
- 编辑、调试程序

对各种功能的访问都设计成直观、易学的方式。

可以使用SIMATIC管理器在下列方式工作：

- 离线方式，不与可编程控制器相联
- 在线方式，与可编程控制器相联

注意相应的安全提示。

如何进行进一步操作

以“Projects”形式建立自动化任务。在操作之前，如果先理解下列内容，你就会感到这个操作变得非常简单：

- 用户接口
- 基本操作步骤
- 在线帮助

5.2 启动 STEP 7 并带有预置启动参数

使用STEP 7 V5.0以上版本，你可以在SIMATIC管理器中生成符号，并在调用顺序上指定启动参数。这样，SIMATIC管理器可以快速定位这些参数描述的对象，双击鼠标即可立即进入相应的Project。

执行文件**S7tgtopx.exe**，可以指定下列起动参数：

/e <完整的物理Project路经>

/o <对象的逻辑路径>

/h <对象ID>

/onl

该启动参数可以在线打开项目指定项目文件

/off

该启动参数可以离线打开项目指定项目文件

/keep

该启动参数可以进行如下操作：

当SIMATIC Manager 是打开的，并且已经有开着的项目文件，可以同时再打开另外一个项目。如果SIMATIC Manager还没有打开，则可以打开在任务存储的中曾经打开过的项目。如果该参数没有定义，则启动时首先关闭已经打开的项目，并不考虑在任务存储的中曾经打开过的项目，仅仅打开指定打开的项目。

下列方法可以指导用户如何简单的设置合适的参数。

用复制和粘贴方式建立参数

按如下步骤进行：

1. 建立访问文件S7tgtopx.exe的路径。这个文件位于安装目录下的 S7bin中 。
2. 打开属性对话框。
3. 选择“Link”选项卡。接下来扩展“Target”下的输入。
4. 在SIMATIC管理器选择相应的对象。
5. 用“CTRL+C”复合键，复制这个对象到文件夹。
6. 将光标移“Link”选项卡中“Target”的末端。
7. 用“CTRL+V”复合键，粘贴文件夹中的内容。
8. 用“OK”关闭对话框。

参数举例：

/e F:\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p /keep

/o "1,8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1"

/h T00112001;129;T00116001;1;T00116101;16e /keep

注意Project路径的结构

在文件系统中Project路径是一个物理路径。完整的逻辑路径如下：

[View ID, online ID]:project name\{object name\}*\ 对象名

例如：/o 1.8:MyConfig\SIMATIC 400（1）\CPU416-1\S7-Program（1）\Blocks\FB1

注意逻辑路径的结构

只能用复制和粘贴功能，建立完整的逻辑路径和对象ID。

但是，用户也可以访问这些路径。如前例所述：

/o "MyConfig\SIMATIC 400（1）\CPU416-1\S7-Program（1）\Blocks\FB1”。通过增加/onl或/off，用户可以指明路径是否在在线窗口或离线窗口有效。如果仅做复制和粘贴则无需做上述工作。

重要：如果路径中有空格，则必须放在引号内。

5.3 访问帮助功能

在线帮助

在线帮助系统提供给用户有效快速的信息，无需查阅手册，在线帮助具有如下信息方式：

- Contents：显示帮助信息的号码。
- Context-Sensitive Help（F1 key）：首先用鼠标选中或在对话框或窗口选择某一对象，而后用F1键得到相应帮助信息。
- Introduction：对某种功能的使用、主要特性及功能范围做一个简要说明。
- Getting Started：概述启动某功能的基本步骤。
- Using Help：在在线帮助下，对查找特殊信息的方法提供描述。
- About：提供有关当前版本的信息。

通过帮助菜单可访问任何窗口的实时对话框。

访问在线帮助功能

你可以用下列方法之一访问在线帮助：

- 在菜单栏选中帮助菜单，再从中选择相应指令。
- 在某对话框单击“Help”键，则可以显示这个对话框的帮助。
- 将光标移到某个你希望得到帮助的窗口或对话框，而后按下F1键或选择菜单命令Help > Context-Sensitive Help。
- 在窗口内使用问号键。

在线帮助功能访问的后三种方式即大家所熟悉的上下文相关帮助。

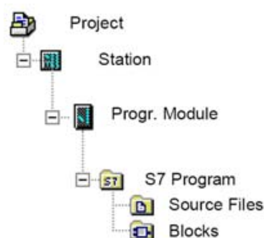
访问快速帮助功能

你只要将光标移到需要帮助的工具栏的某个键上，保持片刻，则显示出快速帮助信息。

5.4 对象和对象层次

与Windows Explorer的文件夹和文件的结构相同，SIMATIC管理器可以在STEP 7的项目和库中显示对象层次。

下图是一个对象等级的例子。



- 项目对象
- 站对象
- 编程模块对象
- S7/M7程序对象
- 源文件文件夹对象
- 块文件夹对象

对象有下列功能：

- 对象属性的载体
- 文件夹
- 功能的载体（如：启动某特殊应用）

对象作为属性的载体

对象同时携带功能和属性（如设定）。当选择某一对象，则能够执行下列功能：

- 用菜单命令**Edit > Open Object**，编辑对象
- 用菜单命令**Edit > Object Properties**，打开对话框，设定对象的特定选择

文件夹也可以做为属性的载体。

对象作为文件夹的功能的载体

文件夹（子目录）可能含有其它的文件夹或对象。当你把它打开后，则可以显示这些内容。

对象作为功能的载体

当你打开一个对象以后，则显示一个可编辑的窗口。

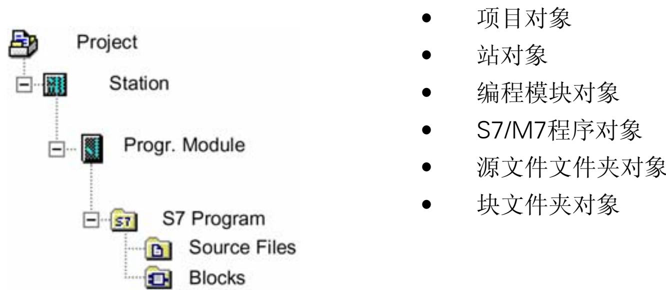
一个对象，或是一个文件夹，或是一个功能的载体。除非是这样一个站：它即是文件夹（编程模式），也是功能载体（用于硬件组态）。

- 当你双击某个站的图标，其中的对象就显示出来：可编程模板和站的组态（站作为一个文件夹）。
- 当你用菜单命令Edit > Open Object打开一个站时，你就能对其进行组态和参数设定（站作为一个功能载体）。当你双击“Hardware”对象时，与菜单命令有着相同的效果。

5.4.1 项目对象

项目位于对象等级最高层，它包括一个自动化解决方案中的所有的数据及程序。

在项目视窗中的位置



- 项目对象
- 站对象
- 编程模块对象
- S7/M7程序对象
- 源文件文件夹对象
- 块文件夹对象

图　标	对象文件夹	重要功能选择
	项目	• 建立项目 • 项目和库的归档 • 管理多语言文本 • 检查可使用的软件包 • 打印项目文件 • 再排列 • 翻译并编辑操作文本 • 插入操作站对象 • 多个用户编辑项目 • 转换版本1 项目 • 转换版本2 项目 • 设置PG/PC接口
	站 SIMATIC 300 SIMATIC 400	• 插入一个站点 • 这个图标即表示一个站也表示一个文件夹(站的等级)，其它功能可以在站对象下找到
S7	S7程序	• S7/M7程序，没有站或CPU • S7/M7程序图标即是对象（项目级），也是对象文件夹（程序级）。其它功能可以在S7/M7程序对象中找到

图　标	对象文件夹	重要功能选择
	M7程序	
	网络 用于网络组态和设定属性的启动工具	• 通讯站点和分支网络的属性 • 总览：整体数据通讯 • 组态整体数据通讯过程

5.4.2　库对象

库中包含S7/M7程序，用于存贮程序块。库位于对象等级的上层。

S7 Program (1) Source Files Blocks	• 库对象 • S7/M7程序对象 • 源文件文件夹对象 • 块文件夹对象

图标	对象文件夹	重要功能选择
	库	• 标准库总览 • 有关程序库 • 项目和库的归档

图标	库级中的对象	重要功能选择
	S7 程序 M7 程序	• 插入S7/M7程序 • S7/M7程序图标即是对象（项目级），也是对象文件夹（程序级）。其它功能可以在S7/M7程序对象中找到

5.4.3　站对象

SIMATIC 300/400站代表了用一个或多个可编程模板组成的S7硬件的组态。

在项目视窗中的位置

Project
Station
Progr. Module
S7 Program
Source Files
Blocks

- 项目对象
- **站对象**
- 编程模块对象
- S7/M7程序对象
- 源文件文件夹对象
- 块文件夹对象

图标	对象文件夹	重要功能选择
	站	• 插入一个站点 • 上载站 • 下载组态参数到可编程控制器 • 从一个站点上载组态参数 • 显示CPU报文和用户定义的诊断报文 • 组态“系统错误报告” • 诊断硬件，显示模板信息 • 显示和改变操作模式 • 显示并设置时间和日期 • 清除装载存储器和工作存储器，初始化CPU
	SIMATIC PC站 (没分配)	• 生成并设定SIMATIC PC 站点参数 • 组态SIMATIC PC站间的连接 • 上载SIMATIC PC站
	SIMATIC PC站 (已分配)	• 在Network View 中组态的SIMATIC PC 站

图标	站点级中的对象	重要功能选择
	硬件	• 硬件组态基本步骤 • 站点组态基本步骤 • 总览：组态、设定本机配置参数的步骤 • 组态DP主站系统基本步骤 • 组态多计算操作
	编程模块	• 编程模块既是对象（站点级）又是对象文件夹（“Programmable Modules”级）。其它功能可以在编程模块对象中找到

5.4.4 编程模块对象

编程模块可以再现设定的参数（CPUxxx，FMxxx，CPxxx）。那些没有保持存储区的模板的系统数据（例如CP441）通过该站的CPU装入。因此，那些没有“系统数据”的对象在项目等级中不能显示。

在项目视窗中的位置



图标	站点级中的对象	重要功能选择
	编程模块	• 总览：组态设定本机配置的参数 • 显示CPU报文和用户定义的诊断报文 • 组态“系统错误报告” • 诊断硬件，显示模板信息 • 通过EPROM存储卡下载 • 设定访问可编程控制器的口令 • 显示强制数值窗口 • 显示和改变操作模式 • 显示并设置时间和日期 • 设定操作属性 • 清除装载存贮区和工作暂存区，初始化CPU • 在线视窗中的诊断符号 • 内存分区 • 将下载软件块存入集成EPROM • 升级可编辑控制器中的操作系统
?	可编程模板的对象	• 显示用更高版本的STEP 7组态的模板

图标	编程模块级对象	重要功能选择
	程序： S7程序 M7程序 程序	• 插入S7/M7程序 • S7/M7程序图标既是对象(Project级)，也是对象文件夹（Program级）。其它功能可以在S7/M7程序对象中找到
	设定网络	• 项目中的网上站点 • 连接类型及站点 • 明确连接的不同类型 • 输入新的连接 • 组态用于SIMATIC站中的模板连接

5.4.5 S7/M7程序对象

S7/M7程序是一个文件夹，它包括 S7/M7 CPU模板的软件或非CPU模板的软件（如：可编程的CP或FM模板）。

在项目视窗中的位置

Project	• 项目对象
Station	• 站对象
Progr. Module	• 编程模块对象
S7 Program	• **S7/M7程序对象**
Source Files	• 源文件文件夹对象
Blocks	• 块文件夹对象

图标	对象文件夹	重要功能选择
	S7程序	• 插入S7-/M7-程序 • 建立地址优先权 • 建立逻辑块的基本步骤 • 分配消息号 • 如何为项目分配和编辑用户定义的诊断信息（基于项目的） • 如何为CPU分配和编辑用户定义的诊断信息（基于CPU的） • 翻译并编辑操作文本 • 管理多语言文本 • 显示CPU报文和用户定义的诊断报文 • 程序测试

图标	对象文件夹	重要功能选择
M7	M7程序	• M7系统程序
	程序	• 一般指在项目中生成的软件块

图标	程序级中的对象	重要功能选择
	对单一或多个变量设定符号的符号表	• 绝对地址和符号地址 • 符号表的结构及元素 • 输入共享符号 • 输入符号时的提示 • 如何为项目分配和编辑用户定义的诊断信息（基于项目的） • 如何为CPU分配和编辑用户定义的诊断信息（基于CPU的） • 翻译并编辑用户文本 • 通过符号表组态控制与监测属性 • 编辑通讯属性 • 符号表输入/输出接口
	源文件	• 其它功能可以在源文件夹对象中找到
	块文件夹	• 其它功能在块文件夹对象中寻找
	文本库文件夹	• 用户文本库

5.4.6 块文件夹对象

离线方式软件块文件夹包括：逻辑软件块（OB，FB，FC，SFB，SFC），数据块（DB），用户定义数据类型（UDT）和变量表。系统数据对象表示系统数据块。

在线方式的块文件夹包括可下载到可编程控制器并可执行的程序。

在项目视窗中的位置



- 项目对象
- 站对象
- 编程模块对象
- S7/M7程序对象
- 源文件文件夹对象
- **块文件夹对象**

图标	对象文件夹	重要功能选择
	块	• 用Project Management（项目管理）下载 • 不用Project Management（项目管理）下载 • 可用参考数据概述 • 再接线 • 块的比较 • 翻译并编辑操作文本 • 转换软件块的系统属性、语言描述及提示

符号	块文件夹对象	重要功能选择
	块概要	• 建立逻辑软件块 • 生成软件块 • 编写STL源文件的基本信息 • 块的比较
	组织块（OB）	附加功能： • 引入数据及参数类型 • 下载条件 • 用程序状态功能进行测试 • 你应了解的关于在单步模式/断点下进行测试的信息 • 再接线 • 块的注释
	功能（FC）	附加功能： • 引入数据及参数类型 • 下载条件 • 用程序状态功能进行测试 • 你应了解的关于在单步模式/断点下进行测试的信息 • 再接线 • 块及参数属性
	功能块（FB）	附加功能： • 引入数据及参数类型 • 使用多重背景 • 下载条件 • 用程序状态功能进行测试 • 你应了解的关于在单步模式/断点下进行测试的信息 • 再接线 • 软件块及参数属性 • 如何设定和编辑与块相关的信息（基于项目的） • 如何创建有关块的信息（基于CPU的） • 如何进行PCS 7 报文组态（基于项目的） • 如何进行PCS 7 报文组态（基于CPU的） • 翻译并编辑用户文本 • 设定FB参数的监视/控制属性

符号	块文件夹对象	重要功能选择
	用户定义的数据类型（UDT）	• 创建块 • 编写STL源文件的基本信息 • 引入数据及参数类型 • 使用用户定义的数据类型访问数据 • 块及参数的属性
	DB（全局数据块）	• 数据块中的数据总览 • 数据块的声明表显示状态 • 下载条件 • 编写数据块状态 • 引入数据及参数类型 • 使用多重背景 • 软件块及参数属性 • 如何设定和编辑块的相关信息（基于项目，只限于背景数据块） • 如何设定和编辑块的相关信息（基于CPU，只限于背景数据块） • 如何组态PCS7报文（基于项目，仅限于背景数据块） • 如何组态PCS7报文（基于CPU，仅限于背景数据块） • 翻译和编辑用户文本（仅限于背景数据块）
	系统功能(SFC)	• 下载条件 • 软件块及参数属性 • 软件块的注释
	系统功能块（SFB）	• 下载条件 • 块及参数属性 • 如何设定和编辑与块相关的信息（基于项目的） • 如何创建有关块的信息（基于CPU的） • 如何进行PCS 7 报文组态（基于项目的） • 如何进行PCS 7 报文组态（基于CPU的） • 翻译并编辑用户文本 • 块的帮助文件
	带KNOW HOW保护的块	• 在STL源文件中定义块属性的规则 • 块属性
	具有诊断功能的块	S7-PDIAG可选软件包手册中有详细介绍
	用F-FBD/-LAD/-STL/-DB编程语言生成的块	S7 Distributed Safety可选软件包手册中有详细介绍
	变量表（VAT）	• 用变量表进行监视和修改的基本程序 • 测试变量表 • 监测变量 • 修正变量 • 强制变量

符号	块文件夹对象	重要功能选择
	系统数据块（SDB）	系统数据块（SDB）仅能用下列功能间接的进行编辑： • 硬件组态 • 通讯站点和分支网络的属性 • 总览：整体数据通讯 • 设定和编辑与符号相关的信息 • 下载条件

5.4.7 源文件文件夹对象

源文件文件夹包含有文本方式的源程序。

在项目视窗中的位置

Project
Station
Progr. Module
S7 Program
Source Files
Blocks

- 项目对象
- 站对象
- 编程模块对象
- S7/M7程序对象
- **源文件文件夹对象**
- 块文件夹对象

图标	对象文件夹	重要功能选择
	源文件夹	• 编写STL源文件的基本信息 • 导出源文件 • 导入源文件

图标	源文件夹对象	重要功能选择
	源文件 (例如：STL源文件)	• 编写STL源文件的基本信息 • 生成STL源文件 • 将块模板插入STL源文件 • 将现有块的源代码插入STL源文件 • 检查STL源文件的一致性 • 编译STL源文件 • 从软件块生成STL源文件 • 导出源文件 • 导入源文件
	LAD指令模板	• 生成LAD指令模板

5.4.8 没有站点或CPU的S7/M7编程

在没有组态SIMATIC站之前，你可以编写程序。这也就意味着初始工作与你要编程的站所使用的模板是相对独立的。

生成S7/M7程序

1. 用菜单命令File > Open或项目窗口，打开相应的项目。
2. 选择离线方式下的项目。
3. 根据在可编程控制器上生成的程序，选择下列菜单命令：
   Insert > Program > S7 Program，如果你的程序要运行在SIMATIC S7设备上。
   Insert > Program > M7 Program，如果你的程序要运行在SIMATIC M7设备上。

这时，S7/M7程序已进入“Project”窗口。它包括一个程序文件夹和一个空的符号表，在此你可以建立，编写程序块。

将程序与模板相对应

当所编写的程序与某种特定的模板无关时，用户可以简单的甚至通过托拽的方式，将这些程序的地址及符号表进行拷贝或移植，从而将程序与模板对应起来。

添加程序到库中

如果你需要多次使用某个SIMATIC S7的程序，类似于“软件池”，则可将这个程序放入库中。但是测试时，一定将其放入项目中，因为这是程序与可编程控制器关联的唯一途径。

访问可编程控制器

选择项目的在线方式。在含有程序属性的对话框中设定地址。

提示

当删除站点或CPU的时候，你会得到是否删除其中程序的提示。如果你选择不删除程序，则这个程序以无站点的形式直接留在项目中。

5.5 用户接口与操作

5.5.1 操作原理

目的：操作简易

图形化的用户接口，使软件的操作更加直观。你可以用日常中熟悉的方式在软件中找到所需对象，例如站点、模板、程序、块。

包括对象的建立、选择及操作，都可以在STEP 7软件平台上完成。

使用不同的工具软件

当使用常规的工具时，还是要首先考虑到需要哪些特定的工具来完成特殊的功能。

以对象为导向的操作的基本流程是确定使用哪种对象，而后打开这个对象，对其进行编辑。

使用对象为导向的操作，无须知道特定的指令。在GUI上，用户只须用菜单命令或光标点击则可以打开对象。

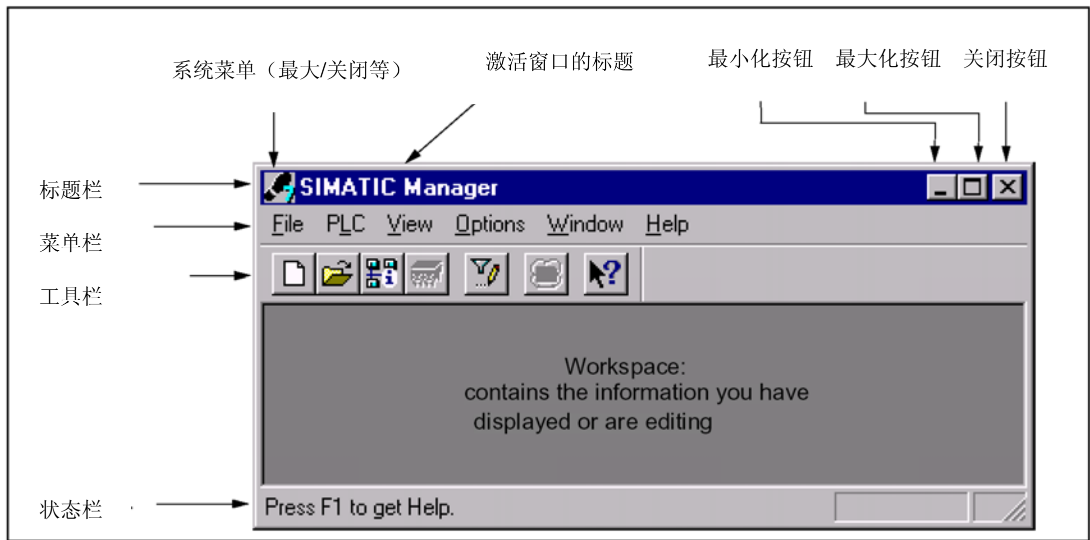
当你打开一个对象以后，相应的显示或编辑该对象的软件会自动启动。

请阅读

下面几页讲述编辑对象的基本步骤。因为在手册中，对此没有详细说明，请用心详读下列部分。

5.5.2 窗口内容排列

标准窗口内容排列如下：



标题栏和菜单栏

标题栏和菜单栏位于窗口的最上端。标题栏包括窗口以及标题和控制窗口的图标。菜单栏包括窗口所有有效菜单。

工具栏

工具栏由图标（或工具按钮）组成，这些图标以快捷键方式作为经常使用的菜单命令用鼠标点击执行。当用光标选中某个快捷键时，在状态栏会有简单的信息显示。

如果某些键不能操作，则呈灰色。

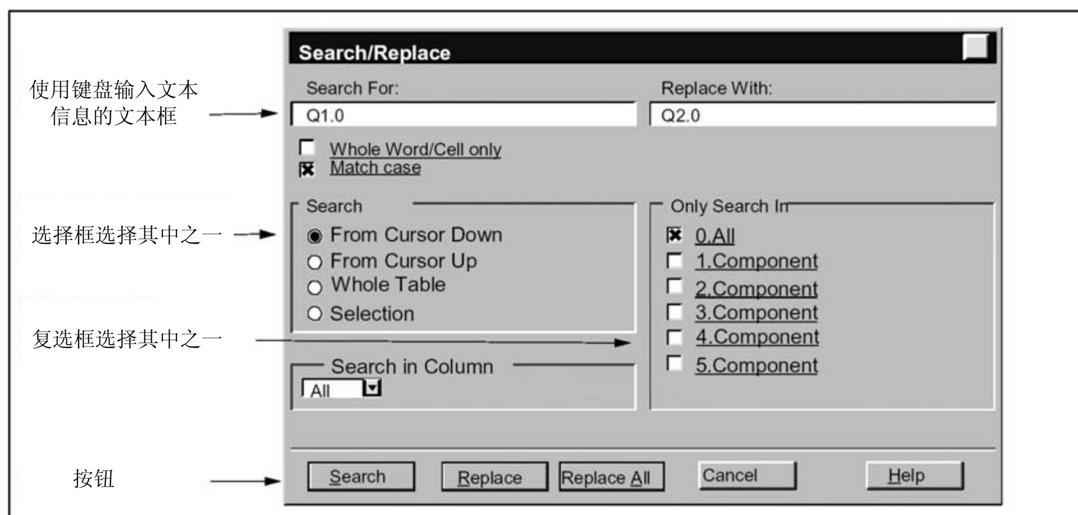
状态栏

状态栏显示相关信息。

5.5.3 对话框中的元素

在对话框中输入信息

你可以将某些特定任务信息输入对话框。经常显示的部分如下图所示：

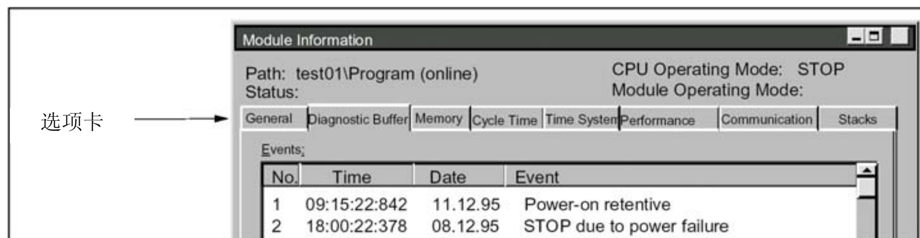


下拉对话框和组合对话框

有时文本框旁边带有一个向下的箭头。这个箭头表示还有更多的选择。点击这个箭头，则显示一个列表框或组合框。如果用光标点击其中之一，则它可以自动显示在文本框中。

对话框中的选项卡

某些对话框由选项卡组成，通过将对话框分为选项卡，来提高信息的清晰度（如下图）。



选项卡出现在对话框的最上端。你只要用光标点击某个选项卡，这个选项卡就会凸出来。

5.5.4 对象的建立和管理

无论哪一种对象基本操作步骤都是相同的，基本操作顺序在手册中另有说明，在此汇总如下：

- 建立对象
- 选择对象
- 在对象执行操作（如复制，删除）

设定建立新的Projects/Libraries的路径

新的用户项目、库和多重项目存储在缺省的文件夹“\Siemens\Step7\S7prog”中。如果要把它们存储在其它路径下，在存储前应先设置该路径。在第一次建立一个新的Project/Libraries之前，用菜单命令**Options>Customize**，用对话框的“General”选项卡，建立你所希望的对象存贮的路径。

建立对象

利用STEP 7 Wizard “New Project”，建立新的项目，并且插入对象。使用菜单命令**File >“New Project” Wizard**，打开Wizard，对话框中将显示你所建立的项目结构，而后利用Wizard建立项目。

如果不用Wizard，菜单命令**File>New**，也能建立项目和库。这些对象形成对象体系的起始部分。你可以用插入菜单建立那些不能自动建立的对象。此外，用硬件组态或“New Project” Wizard，在SIMATIC站建立模板对象。

打开对象

打开对象有多种方式，如下所示：

- 双击对象图标
- 选择对象及菜单命令**Edit > Open Object**。这只适用于文件夹中没有的对象。

一旦打开对象，你就能生成或改变它的内容。

当打开一个不含有其它对象的一个对象时，可生成一个新的窗口，对其内容进行编辑。当其正在被其他应用调用时，你不能改变这个对象。

提示

例外：站点以文件夹的方式显示可编程模块（当你双击它们）和站点组态。当你双击“Hardware”对象，硬件组态功能开始启动。这和选择菜单命令**Edit > Open Object**效果相同。

建立对象体系

用“New Project”Wizard建立对象体系。打开一个文件夹，它所包括的对象也就显示出来。这时你可以用插入菜单建立更多的对象，如：在Project中增加站点。只有那些对于该对象可以插入到当前文件夹的命令，才能在“插入”菜单中被激活。

设定对象属性

对象属性是指对象本身的一些数据，这些数据决定对象的性能特点。当建立一个新的对象，设定对象属性的对话框会自动显示，用于对象属性的设定。对象属性也可以后期改变。

调用菜单命令Edit>Object Properties，打开对话框或设定对象属性。

调用菜单命令Edit > Special Object Properties，打开对话框，输入有关操作控制，监测功能及组态信息。

例如：为了显示一个块的特殊对象关于操作控制及监测功能属性，这个块必须置有相应的操作控制及监测功能的符号，也就是将系统属性“S7_m_c”的属性“Attribute”键设为“true”。

提示

- “System Data”文件夹和“Hardware”对象的属性不能显示或改变。
- 不能在“只读”Project对话框中输入数据。这时输入框为灰色。
- 当显示编程模块时，由于一致性的原因，你这时不能编辑这些显示的参数。想要编辑这些参数，应调用“Configuring Hardware”功能。
- 如果用户仅仅在自己的编程设备上改变了一些对象的设置（例如：模板的参数），它们是不会对实际设备有影响的，因为这些系统数据必须要下载到实际系统中才会起作用。
- 如果用户下载了整个用户程序，则系统数据也将被自动下载。因而如果用户此时再修改设置，则须重新下载系统数据“System data”。

剪切、粘贴、复制

所有的对象都可以在窗口下剪切、粘贴、复制。也可以在编辑菜单中找到这类指令。

你也可以用光标拖、放的方法，复制对象。如果错误地将其移动或复制到一个非法的目标，光标就会显示一个禁止的报警信号。

当复制某个对象的时候，这个对象体系下面的所有内容都将被复制。这样所建立在该对象的内容就可以多次使用。

提示

“Connection”文件夹中的相关连接表不能被复制。注意这时你所复制的操作相关文本表，目标对象仅能接受它已有的语言形式。

在复制对象时，将会有一步一步地提示。

对象更名

SIMATIC管理器为一些新的对象设计有一些标准名。一般来讲这些名字是由对象的类型（如果在相同的文件夹中也可以生成同样类型的对象代码）和代码组成。

例如：第一个S7程序名为“S7 Program（1）”，第二个名为“S7 Program（2）”等。符号表名为“Symbols”，因为它在每一个文件夹中仅存在一个。

你也可以改变对象和Project的名字，使其新名字与内容相关。

同Project一样，路径名不能超过8个字符。否则，当压缩归档和使用“C for M7”（Borland编译器）时，就会出现问题。

你可以直接地或用对象本身属性改变对象的名字。

用直接方式：

- 用鼠标慢慢的点击对象名两次，这时对象名边框就会显示出来。你再用键盘输入一个新的名字。

用菜单：

- 在项目菜单中选择相应对象及菜单命令**Edit > Rename**。这时对象名边框就会显示出来。你再用键盘输入一个新的名字。

如果某个对象名字不能修改，则该对话框的输入区域呈灰色，显示当前名字，不能输入文本信息。

提示

当编辑名字或执行其它操作（例如：选择菜单命令）时，如果你将光标移到对象名框之外，这个操作就会结束。但修改的新名字还是可以被接受。

在“Renaming Objects”菜单下，会有详细的提示。

移动对象

你可以利用SIMATIC管理器，将对象从一个文件夹移到甚至另一个不同Project的文件夹下。当移动一个文件夹的时候，它的所有内容都同时被移走。

提示

下列对象不能移动：

- 连接
- 在线显示的系统数据块（SDB）
- 在线显示的系统功能（SFC）和系统功能块（SFB）

在移动对象时，在“Moving Objects”下会有详细提示。

对象分类

依据对象属性对其分类预览（菜单命令**View > Details**）。用鼠标点击相关的属性的标题。当再次点击时，对象的顺序就会反过来，块由它们自身的数码顺序分类，例如：FB1、FB2、FB11、FB12、FB21、FC1。

对象预置分类

当再次打开Project时，将根据预置分类次序显示对象。如：

- 软件块显示依次为：“系统数据、OB、FB、FC、DB、UDT、VAT、SFB、SFC。”
- 在Project里面，首先显示所有的站，而后是S7 程序。

因此，预置分类不能以字母增减的顺序详细预览。

预置分类顺序的恢复

例如：再分类后，点击标题“Object Name”上端，就能对预置分类进行恢复，具体操作顺序如下：

- 在详细预览时点击“Type”。
- 关闭Project，而后再打开。

删除对象

你可以删除一个文件夹及对象。当删除一个文件夹的时候，其中的所有对象也同时被删除。

你不能取消删除操作。当你不能确认某个对象是否保留，最好首先将整个Project存档。

提示

下列对象不能删除：

- 连接
- 在线显示的系统数据块（SDB）
- 在线显示的系统功能（SFC）和系统功能块（SFB）

在删除对象时，在“Deleting Object”下会有详细提示。

5.5.5 在对话框中选择对象

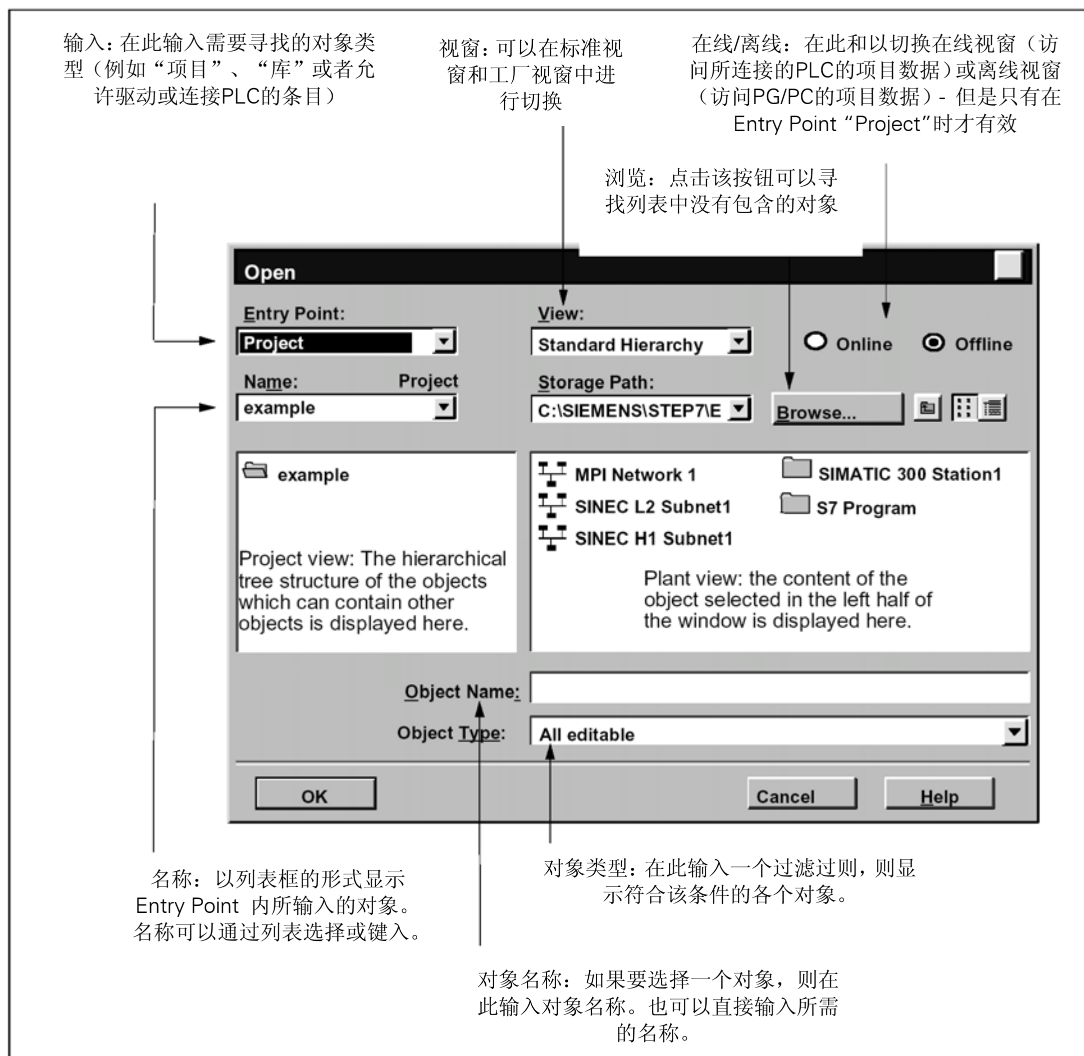
用浏览器对话框选择对象，一般需要进行多步骤的操作。

调用浏览器

你可以用硬件组态操作方式调用浏览器对话框，如：用菜单命令**Station>New /Open**（“SIMATIC Manager”窗口除外）。

浏览器对话框的结构

在对话框中有下列部分可供你选择。



5.5.6 任务记忆存贮器

SIMATIC管理器可以存贮窗口的内容（打开的Project和Libraries）及排列方式。

- 你可以用菜单命令Options > Customize，决定在任务结束时是否存贮窗口内容及排列。下一个任务开始时恢复这些内容。在打开的Project画面上，光标位于最后选择的文件夹。

- 用菜单命令Window > Save Settings，存贮实时运行的窗口内容及排列。
- 你可以用Window > Restore Settings命令，恢复用Window > Save Settings存贮的窗口内容及排列方式。在打开的“Project”画面上，光标位于最后选择的文件夹。

提示

在线“Project”的窗口内容、“Accessible Nodes”的窗口内容和“S7 Memory Card”的窗口内容不能保存。访问PLC（S7-300/S7-400）的口令，在任务结束时不能存贮。

5.5.7 改变窗口的排列

让窗口重叠放置并能看见每个窗口的标题，选择如下操作：

- 菜单命令Window > Arrange > Cascade
- 按SHIFT + F5组合键

如果你想从上到下排列所有窗口，选择菜单命令Window > Arrange > Horizontally。

如果你想从左到右排列所有窗口，选择菜单命令Window > Arrange > Vertically。

5.5.8 窗口排列的存贮及恢复

STEP 7具有这样的特性：存贮实时窗口的排列。在下一次显示时按存贮的排列方式显示窗口。可以使用菜单命令Options > Customize下“General”选项卡完成。

存贮什么内容?

当存贮窗口排列时，存贮下列信息：

- 主窗口的位置
- 打开的项目和库及它们的窗口位置
- 窗口级联的次序

提示

在线“Project”窗口，“Accessible Nodes”窗口和“S7 Memory Card”窗口不能存贮。

存贮窗口排列

用菜单命令Window > Save Settings，保存当前窗口排列。

恢复窗口排列

用菜单命令Window>Restore Settings，恢复窗口排列。

提示

当恢复窗口时，只能显示已存贮的对象层的内容。

5.6 键盘控制

国际键名	德国键名
HOME	POS 1
END	ENDE
PAGE UP	BILD AUF
PAGE DOWN	BILD AB
CTRL	STRG
ENTER	Eingabetaste
DEL	ENTF
INSERT	EINFG

5.6.1 用于菜单命令的组合键

用ALT键和组其它键组合，可任意选择菜单命令。

依次按下例键：

- ALT键
- 菜单中带下划线的字符（如：ALT，F用于“File”），如果菜单栏中含“File”的话。这个菜单就会打开。
- 菜单命令中带有下划线的字符（如ALT， N用于“New”菜单命令）。如果这个菜单命令中含有子菜单，子菜单也同时打开。如上述方式可以选择所有菜单命令。

一旦输入组合健的最后一个字符，菜单命令就开始执行。

例如：

菜单命令	组合键
File > Archive	ALT，F，A
Window > Arrange > Cascade	ALT，W，A，C

菜单命令快捷键

命令	快捷键
New （“File” Menu）	CTRL+N
Open （“File” Menu）	CTRL+O
Save as （“File” Menu）	CTRL+S
Print > Object Table （“File” Menu）	CTRL+P
Print > Object Content （“File” Menu）	CTRL+ALT+P
Exit （“File” Menu）	ALT+F4
Cut （“Edit” Menu）	CTRL+X
Copy （“Edit” Menu）	CTRL+C
Paste （“Edit” Menu）	CTRL+V
Delete （“Edit” Menu）	DEL
Select All （“Edit” Menu）	CTRL+A
Rename （“Edit” Menu）	F2
Object Properties （“Edit” Menu）	ALT+RETURN
Open Object （“Edit” Menu）	CTRL+ALT+O
Compile （“Edit” Menu）	CTRL+B
Download （“PLC” Menu）	CTRL+L
Diagnostics/Setting > Module Status （“PLC” Menu）	CTRL+D
Diagnostics/Setting > Operating Mode （“PLC” Menu）	CTRL+I
Update （“View” Menu）	F5
Updates the status display of the visible CPUs in the online view	CTRL+F5
Customize （“Options” Menu）	CTRL+ALT+E
Reference Data > Display （“Options” Menu）	CTRL+ALT+R
Arrange > Cascade （“Window” Menu）	SHIFT+F5
Arrange > Horizontally （“Window” Menu）	SHIFT+F2
Arrange > Vertically （“Window” Menu）	SHIFT+F3
Context-Sensitive Help （“Help” Menu）	F1 （如果选择菜单命令，则显示其提示内容。否则只显示提示起始画面）

5.6.2 光标移动组合键

在菜单栏/弹出菜单中移动光标

功能	按键
移到菜单栏	F10
移到弹出菜单	SHIFT+F10
移到带有下划线字符或数字的菜单	ALT+菜单中带有下划线的字符
选择带有下划线字符的命令	菜命令中带有下划线的字符
移向左边菜单命令	左箭头
移向右边菜单命令	右箭头
移向上边菜单命令	上箭头
移向下边菜单命令	下箭头
执行菜单命令	ENTER键
放弃菜单名或关闭打开的菜单并返回	ESC键

编辑文本时光标的移动

功能	按键
向上移一行或如文本仅有一行向左移一个字符	上箭头
向下移一行或如文本仅有一行向右移一个字符	下箭头
左移一个字符	左箭头
右移一个字符	右箭头
左移一个字	左箭头
右移一个字	右箭头
移到行首	HOME
移到行尾	END
翻转到上一页	PAGE UP
翻转到下一页	PAGE DOWN
移到文本起始位位置	CTRL+HOME
移到文本起终止位置	CTRL+END

在对话框中移动光标

功能	按键
从一个对话框到另一个对话框（从左到右，从上到下）	TAB
在对话框内以反方向移动	SHIFT+TAB
选择带下划线字符菜单	ALT+菜单中带有下划线的字符
选择选项列表	箭头键
打开选项列表	ALT+下箭头键
选择或放弃菜项标题	空格键
确认输入并关闭对话框（“OK”键）	ENTER
关闭对话框对修正不做存贮（“Cancel” 按钮）	ESC

选择文本的组合键

选择或放弃文本	按键
右移一个字符	SHIFT+右箭头键
左移一个字符	SHIFT+左箭头键
移到行的起始位置	SHIFT+HOME
移到行的终止位置	SHIFT+END
移到上一行	SHIFT+上箭头键
移到下一行	SHIFT+下箭头键
翻转到上一页	SHIFT+PAGE UP
翻转到下一页	SHIFT+PAGE DOWN
移到文本文件起始位位置	CTRL+SHIFT+HOME
移到文本文件起终止位置	CTRL+SHIFT+END

5.6.3 访问在线帮助的组合键

功能	按键
打开帮助功能	F1（如正在执行菜单命令，则显示相应帮助否则显示功能初始画面）
选择带有问号的帮助键	SHIFT+F1
关闭帮助窗口并返回	SHIFT+F4

5.6.4 用复合键完成窗口切换

功能	按键
选择不同的窗口	F6
如没有相应窗口，返回前一级窗口	SHIFT+F6
注释窗口与保留窗口（变量表窗口间的切换）。如果没有保留窗口，则返回前一级窗口	SHIFT+F6
注释窗口间的切换	CTRL+F6
返回到前一级注释窗口	SHIFT+CTRL+F6
在注释窗口间的切换（应用系统和保留窗口），当返回这一系统后则激活最后打开的注释窗口	ATL+F6
返回前一级注释窗口	SHIFT+ALT+F6
关闭时实运行窗口	CTRL+F4

6 创建并编辑项目

6.1 项目结构

项目可用来存储为自动化任务解决方案而生成的数据和程序。这些数据被收集在一个项目下，包括：

- 硬件结构的组态数据及模板参数
- 网络通讯的组态数据
- 为可编程模板编制的程序

生成一个项目的主要任务就是为编程准备这些数据。

数据在一个项目中以对象的形式存储。这些对象在一个项目下按树状结构分布（项目层次）。在项目窗口中各层次的显示与Windows资源管理器中的相似。只是对象图标不同。

项目层次的顶端结构如下：

1层：项目

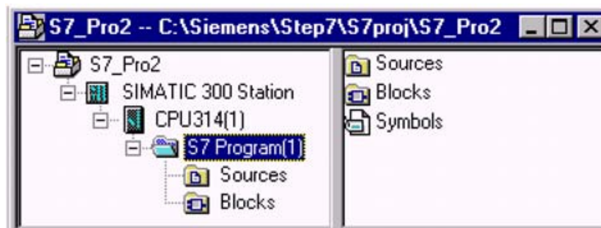
2层：子网、站或S7/M7程序

3层：依据第二层中的对象而定

项目窗口

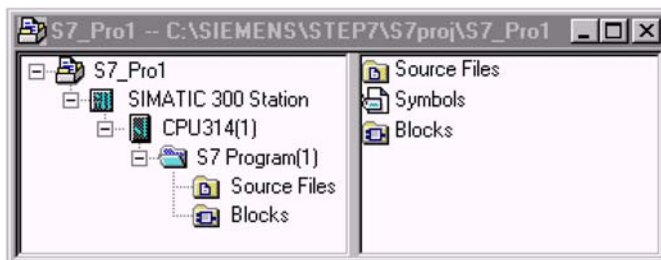
项目窗口分成二个部分。左半部显示项目的树状结构。右半部窗口显示左半窗口中打开的对象中所包含的各个对象，显示方式选择（大符号，小符号，列表或详细）。

在左半窗口点击“+”符号以显示项目的完整的树状结构。最后的结构看起来就象下图一样。



在对象层次的顶层是对象“S7_Pro1”作为整个项目的图标。它可以用来显示项目特性并以文件夹的形式服务于网络（组态网络）、站（组态硬件）以及S7或M7程序（生成程序）。当选中项目图标时，项目中的对象显示在项目窗口的右半部分。位于对象层次（库以及项目）顶部的对象在对话框中形成一个起始点用以选择对象。

项目查看



在项目窗口中，你可以通过选择“offline（离线）”显示编程器中该项目结构下已有的数据，也可以通过选择“online（在线）”显示可编程控制系统中已有的数据。

如果安装了相应的可选软件包，你还可以设置另外一种查看方式：设备查看。

注意

硬件及网络的组态只能在“offline”下进行。

6.2 了解访问保护

在STEP7 V5.4中，用户可以通过设置密码来限制对项目和库的访问。当然，首先需要安装“SIMATIC Logon”。

用户也可以使能、中止以及显示改变的日值。

安装了“SIMATIC Logon”后，在SIMATIC Manager中将出现下列一些指令，用户可以通过这些指令来管理访问项目和库文件。

- Access Protection, Enable 访问保护，使能
- Access Protection, Disable 访问保护，禁止
- Access Protection, Manage Users 访问保护，管理用户
- Access Protection, Adjust in Multiproject 访问保护，在多个项目中调整
- Remove Access Protection und Change Log 去除访问保护以及改变登陆日志

在SIMATIC Manager下，选择Options > Access Protection, Enable指令，在打开的对话框中可以为项目分配一个密码，下次再打开项目或库进行编辑时，就需要输入该密码。

通过菜单上**去除访问保护以及改变登陆日志**（Remove Access Protection und Change Log）指令可以去除保护，然后可以通过 STEP7 V5.4 再进行编辑。

注意

- 为了能够使能或者禁止保护功能，用户自己必须在 SIMATIC Logon 中被授权为项目管理者。
- 首次使能保护功能时，项目的格式会发生变化。用户会看到一条信息，提示该项目不能在被以前的旧版本的STEP7进行编辑。
- 通过去除保护功能的指令，用户可以再用STEP7 V5.4 对项目进行编辑，但会丢失所有曾经改变过项目或库文件的访问者的信息日志。
- 当前的登陆用户名将被显示在SIMATIC Manager的状态栏中。
- 当前的登陆用户可以使能保护功能，是项目的管理员，可以设置密码。
- 打开一个有保护的项目，用户必须是被SIMATIC Longon授权的项目管理员或编辑者。保护功能必须已被启动，用户须知道密码。

6.3 了解项目日志

在STEP7 V5.4中，设置了项目保护功能后，用户可以保存登陆日志，记录在线用户。

例如：

- 激活/取消/组态保护和登陆日志
- 打开/关闭项目和库
- 下载到PLC（系统数据）
- 转载或拷贝块
- 改变操作模式
- 清除/复位

用户可以显示项目日志，并向其中加以注释，例如做过某些改动的原因。

改变日志文件，在菜单**Options >Change Log, Enable**下先使能，然后可以通过**Options > Change Log, Display**显示该文件，当然，也可以通过**Options >Change Log, Disable**来禁止该功能。

注意

- **Options > Access Protection > Remove Access Protection and Change Log** 指令允许项目或库再一次被STEP7 V5.4编辑，但之前的记录用户访问的日志文件将丢失。
- 执行该功能时，必须是被项目管理员在“SIMATIC Logon”中的授权用户，而且保护功能必须已启动。

6.4 使用多语言字符集

在STEP 7 V5.4中，可以输入不同于软件语言设置的其他语言文本。但此时，该语言应与操作系统语言设置相吻合。比如，这样用户就可以在中文版的操作系统上使用英文版的STEP 7，且在STEP 7中输入中文字符。须区分如下语音设置：

Windows 语言设置

该选项在Windows的控制面板中。在语言选择中显示了操作系统语言，用户可以选择多语言。

项目语言

当项目刚被建立时，项目语言与操作系统在控制面板中设置的语言相同。一旦选择，项目语言将无法改变。然而，通过选择"language-neutral"，用户仍可以在不同语言的操作操作系统中打开项目文件。但在改变项目语言为"language-neutral"之前，要保证之前的项目语言必须是英文字符集（ASCII characters 0x2a - 0x7f）。

项目语言可以通过Edit > Object Properties 来查看。在所显示的对话框中，用户可以选择。

可以在任意Windows语言下打开项目，即"language neutral" 。

如果用户是通过菜单命令Save As来拷贝项目，而且项目语言设置与操作系统语音设置不同，仍可在复制的项目中修改语言与操作系统相同。这样，用户可以为项目生成不同语言版本的文件，但首先要保证原项目只使用了英文字符集（ASCII characters 0x2a - 0x7f）。这样，将最大程度的保证项目在不同编辑环境下的数据一致性。

STEP 7 语言

STEP 7的语言可以通过**Option>Customize**来选择，所有接口元素，菜单指令以及对话和错误信息均使用所选择的语言。

如果用户的操作系统选择其他语言比如德语，法语，意大利或西班牙语，只要STEP 7选择英语就都可以在这些操作系统上正常显示。

规则

- 如果用户的操作系统与所编辑的项目或库语言设置不同，请注意以下规则以避免数据不兼容。
- 安装STEP7时，将操作系统设为英文字符（ASCII character 0x2a-0x7f）。
- 项目名称以及安装路径均使用英文。否则，假如用户使用中文名，则该项目只能在中文系统下正常显示。
- 在多重项目中，使用相同的语言，或者其中一个项目使用中立语言设置（languge-neutral）。且该多重项目为中立语言设置。
- 创建库文件时，最好使用中立语言设置，这样就可以保证在各种语言环境下都可以使用。另外库文件名以及各种符号名最好使用ASCII码（ASCII character 0x2a-0x7f）。

- 当导入/导出硬件组态或符号表时，一定要导入/导出语言的兼容文件。
- 用户自定义的名称，使用英文字符（ASCII character 0x2a-0x7f）。
- 在STL源文件中，如果没有使用英文字符来编辑标题、用户或组的话，请用单引号进行标注。

注意

- 如果用户的项目或库的是中立语言的（language-neutral），但与目前的操作系统语言不一致且没有英文字符集，还是有可能造成数据的不兼容，因而建议用户在打开文件前检查是否是使用相同的语言设置。
- 如果用户导出硬件组态或符号表时，最好使用英文字符集，不要使用其他语言设置，这样，导入该文件的操作系统将不会出现字符冲突。
- 导出的使用其他语言（例如德语）的硬件组态或符号表有时只能被相同语言设置的操作系统导入。
- 如果符号表使用了操作系统没有的语言字符，则有可能导致错误的发生。
- 符号名和符号地址须用引号标注（“符号名”）。

基本程序

在项目中使用其他语言的方法：

1. 在控制面板中， 设置为所需要的语言。
2. 创建项目。
3. 按照所设定语言输入字符。

之前生成的项目中，如果还未进行语言设定“not yet specified”，可以通过 Edit>Object Properties 菜单指令将语言设定为当前操作系统使用的语言。

6.5 设置操作系统语言

设置Windows XP以及Windows Server 2003系统语言

1. 如果程序不支持统一字符编码（Unicode），则选择如下设定：
   Control Panel > Regional and Language Options > Advanced > Language for non-Unicode programs。
2. 设置标准区域语言设置：
   Control Panel > Regional and Language Options > Languages > Details 以及 Control Panel > Regional and Language Options > Regional Settings (Standards and Formats)。
   之后可以进行正确的语言输入。

设置Windows 2000系统语言

1. 选择显示所需语言：
   Control Panel > Regional Settings > General > Your locale (location)。
2. 选择所需语言：
   Control Panel > Regional Settings > General > Set as Default。
3. 设置标准区域语言设置：
   Control Panel > Regional Settings > Input Locales。
   之后可以进行正确的语言输入。

6.6 建立一个项目

6.6.1 建立项目

使用项目管理结构来构造一个自动化任务解决方案，你需要生成一个新的项目。新项目应生成在你的“General”菜单中为项目设定的路径下，该操作可通过菜单命令Options > Customize选中。

注意

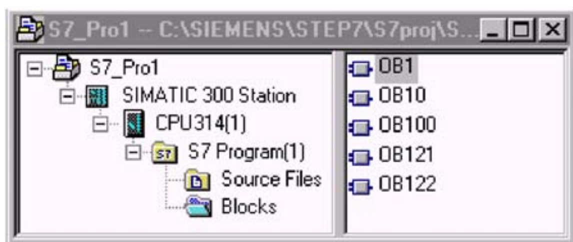
SIMATIC管理器允许项目名多于八个字符。但是，由于在项目目录中名字被截短为八个字符，所以一个项目名的前八个字符应区别于其它的项目。名字不必区分大小写。

无论是手动生成项目还是使用向导（Wizard）生成项目，你都会找到每一步骤的向导。

使用向导生成一个项目

生成一个新项目最简单的办法是使用“New Project（新项目）”向导。使用菜单命令File > “New Project” Wizard，打开Wizard，对话框中将显示你所建立的项目结构，向导会提示你在对话框中输入所要求的详细内容，然后生成项目。除了站、CPU、程序文件夹、源文件夹、块文件夹及OB1外，你甚至还可以选择已有的OB作故障和过程报警的处理。

下图所示为使用向导生成的项目



手动生成项目

你还可以在SIMATIC管理器中，使用菜单命令**File > New**，生成一个新的项目。它已包括“MPI Subnet（MPI网络）”对象。

可选步骤

当你编辑项目时，大部分任务的执行顺序是可以灵活掌握的。一旦生成了一个项目，接下来你可以选择以下的任一方法：

- 首先组态硬件，然后为它生成软件程序，或
- 先生成一个与组态的硬件无关的软件程序

可选方法1：先组态硬件

如果你想先组态硬件，可参照《STEP 7手册》第二卷中组态硬件部分进行。组态硬件完成后，生成软件所需的“S7 Program”和“M7 Program”文件夹则已插入。接下来，继续插入编程所需的对象。最后进行编程。

可选方法2：先生成软件

你可以在没有硬件组态的情况下先生成软件；然后再组态硬件。对于程序编辑来说，并不需要将站的硬件结构事先设好。

基本步骤如下：

1. 在项目中插入没有指定站或CPU 的S7软件文件夹S7/M7程序。

   在这儿你可以决定是否在程序文件夹中包含S7硬件或M7硬件。
2. 然后就可以为可编程模板生成软件了。
3. 组态硬件。
4. 一旦完成硬件组态，就可以将M7或S7程序与CPU联系起来。

6.6.2 插入一个站点

在项目中，站点代表着可编程控制器的硬件结构，它包含着每一个模板的组态数据及参数赋值。

用“New Project（新建项目）”向导生成的新项目中已经包含了一个站。另外，你可以用菜单命令**Insert > Station**生成站。

你可在下列各种站中作选择：

- SIMATIC 300站
- SIMATIC 400站
- SIMATIC H 站
- SIMATIC PC 站
- PC/Programming device（PC/编程器）

- SIMATIC S5
- 其它站，即，非SIMATIC S7/M7及SIMATIC S5

站在插入时带有预置名（如，SIMATIC 300 Station（1），SIMATIC 300 Station（2）等）。如果愿意的话，你可以用一个相应的站名替代预置名。

在帮助“Inserting a Station（插入一个站）”下面，你可以找到一步步插入一个站的向导。

组态硬件

当你组态硬件时，可以借助于模板列表对可编程控制器中的CPU及各模板进行定义。你可以通过双击站来启动硬件组态的应用程序。

一旦你存储并退出硬件组态，对于在组态中生成的每一个可编程模板，都会自动生成S7或M7程序及连接表（“Connections”对象）。用“New Project”向导生成的项目则已包含这些对象。

在帮助“Configuring the Hardware（组态硬件）”下面，你能够找到一步一步组态的向导，更详细的信息见帮助“Basic Steps for Configuring a Station（组态站的基本步骤）”。

生成连接表

每一个可编程模板可自动生成一个（空）的连接表（“Connections”对象）。连接表可用来定义网络中可编程模板之间的通讯连接。打开连接表，则有一个表格窗口显示出来，你可以在这里定义可编程模板之间的连接。

在帮助“Networking Stations within a Project（在一个项目中连网各站）”下面，你可以得到更详细的信息。

下一步骤

一旦完成了硬件组态，你可以为可编程模板编制程序（见帮助“Inserting a S7/M7 Program（插入S7/M7程序）”）。

6.6.3 插入S7/M7程序

为可编程模板编制的软件存储在对象文件夹中。对SIMATIC S7模板而言，该对象文件夹称作“S7 Program”，对SIMATIC M7模板，它则被称作“M7 Program”。

下图是在一个SIMATIC 300站中可编程模板的S7程序的示例。



已存在的组件

每个可编程模板都会自动生成一个S7/M7程序来存储软件；

在新生成的S7程序中，以下对象已经存在：

- Symbol table符号表（“Symbols”对象）
- “Blocks（块）”文件夹，用于存储第一个块
- “Source Files（源文件）”文件夹，用于生成源文件

在新生成的M7程序中，以下对象已经存在：

- Symbol table符号表（“Symbols”对象）
- “Blocks”文件夹

生成S7块

要用语句表、梯形图、或功能块图生成程序，可选择已经存在的“Blocks”对象，然后选择菜单命令**Insert > S7 Block**。在子菜单中，你可以选择想要生成的块的类型（如：数据块、用户定义的数据类型（UDT）、功能、功能块、组织块或变量表）。

你可以打开一个（空）的块，然后用语句表、梯形图或功能图输入程序。你可以在语句表、梯形图及功能图手册里，在生成逻辑块的基本操作的章节中得到更多的相关信息。

注意

在用户程序中，可能会有由系统生成的“系统数据”（SDB）。你可以打开它，但为了保持一致性，你不能修改它。通常当你装载了程序后，再对组态进行了修改，则将SDB下载到可编程控制器就会改变组态。

使用标准库中的块

你可以使用软件提供的标准库中的块来生成用户程序。使用菜单命令**File>Open**，可以访问库。你可以在“Working with Libraries（使用库进行工作）”以及在线帮助中得到更多的有关使用库及生成自己的库的信息。

生成源文件/CFC图表

如果你想用某种特定的编程语言生成一个源文件或CFC图表，可选择S7程序中的对象“Source Files”或“Charts”，然后选择菜单命令**Insert > S7 Software**。在子菜单中选择与你的编程语言相配的源文件。现在可以打开一个（空）的源文件输入程序了。你可以在“Basic Information on Programming in STL Source Files（STL源文件的基本编程信息）”中获得更多的信息。

生成M7程序

如果你想为M7系列的可编程模板的RMOS操作系统编制程序，可选择M7程序。然后选择菜单命令**Insert > M7 Software**。在子菜单中，可选择与你的编程语言或操作系统相匹配的对象。现在，你可以打开所生成的对象并访问相应的编程环境。

生成符号表

当生成一个S7/M7程序时会自动生成一个（空）符号表（“Symbols”对象）。打开符号表时，“符号编辑器”窗口将显示一张符号表，可在该表中定义符号。你可以在“Entering Multiple Shared Symbols in the Symbol Table（在符号表中输入多个共享符号）”中得到更多的信息。

插入外部源文件

你可以用任何ASCII编辑器生成并编辑源文件。然后将这些文件导入到项目中，并且编译生成各个块。

将引入的源文件进行编译，所生成的块存储在“Blocks”文件夹中。

你将在“Inserting External Source Files（插入外部源文件）”中获得更多的信息。

6.7 编辑项目

打开一个项目

要打开一个已存在的项目，可选择菜单命令File > Open，在随后的对话框中选中一个项目，然后，该项目窗口就打开了。

注意

如果你想要的项目没有显示在项目列表中，点击“Browse”按钮。在这里可以搜寻包括已列在项目列表中的项目在内的所有项目。可以使用菜单命令File > Manage改变选项。

拷贝一个项目

使用菜单命令File > Save As，可以将一个项目存为另一个名字。

你可以使用菜单命令Edit > Copy，拷贝项目的某些部分如：站，程序，块等。

你可以在“Copying a Project and Copying Part of a Project（拷贝项目及项目的一部分）”中找到拷贝项目操作的向导。

删除一个项目

使用菜单命令File > Delete，可删除一个项目。使用菜单命令Edit > Delete，可删除项目中的一部分，比如：站，程序，块等。你可以在“Deleting a Project and Deleting Part of a Project（删除项目及删除项目的一部分）”中找到删除项目的操作步骤。

6.7.1 检查项目所使用的选件包

如果你正在编辑的项目中含有选件包所创建的对象，则编辑时需要该选件包。

不论使用什么编程器编辑多重项目、项目或库，STEP 7将提示你所需的选件包及版本。

在下列条件下需要选件包信息：

- 如果用STEP 7 V5.2创建项目(或多重项目中所有项目)或库。
- 首先进入SIMATIC管理器并选择相应项目。选择菜单命令**Edit > Object Properties**。在显示的对话框中，选择“Required optional packages”，该选择框中的信息将告诉你是否检查项目的选件包。

6.8 管理多语言文本

使用STEP 7，可以导出使用一种语言在项目中生成的文本，并进行翻译，重新导入，并以所翻译的语言显示。

下列文本类型可以进行多语言管理。

- 注释和标题
    - 块标题和块注释
    - 网络标题和网络注释
    - STL程序行注释
    - 符号表、变量声明表、用户定义数据类型和数据块注释
    - HiGraph程序注释、状态名和转移名
    - S7-Graph程序中的步名和步注释扩展
- 显示文本
    - STEP 7、S7-Graph、S7-HiGraph或S7- PDIAG生成的报文文本
    - 系统文本库
    - 用户定义的文本库
    - 操作员相关的文本
    - 用户文本

导出

在所选对象中，导出所有块和符号表。每个文本类型都将生成一个导出文件。该文件包括一列源语言和一列目标语言。源语言中的文本禁止改变。

导入

在导入过程中，目标语言栏的内容（右列）将装入所选项目中。只有与找到的源语言栏中现有文本相匹配的文本，才能被接受。

转换语言

当导入项目时，可以选择导入语言。对于所选对象，可以改变“标题和注释”，对于整个项目对象，“显示文本”都可以被改变。

删除语言

在删除一种语言时，该语言中的所有文本都将被从内部数据库中删除。

在项目中应总有一种语言作为基本语言。例如，可以是你的母语。该语言不能删除。在导入和导出时，一定要指定该基本语言为源语言。目标语言可以根据需要设定。

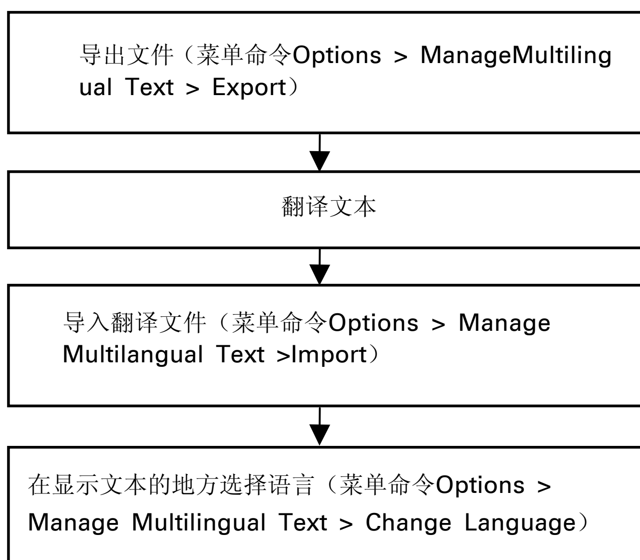
重新组织

重新组织期间，语言变换为当前的语言设置。当前语言设置是在“Language for future blocks”中选择的语言。重新组织只影响标题和注释。

注释管理

你可以制定如何在项目中对块注释进行管理，也就是说以多语言的文本进行管理。

基本程序



6.8.1 多语言文本的类型

导出时每种文本类型将创建一个单独的文件。项目中可以翻译的文本分为以下文本类型：

文本类型	说　明
Block Title	块标题
BlockComment	块注释
NetworkTitle	程序段标题
NetworkComment	程序段注释
LineComment	STL中行注释
InterfaceComment	变量部分注释(程序中的声明表)、UDT注释(用户自定义数据类型)和数据块注释
SymbolComment	符号注释
HiGraphStateName HiGraphStateComment HiGraphTansitionName HiGraphTransitionComment	S7-HiGraph 语句名 语句注释 转换名 转换注释
S7GraphStateName S7GraphStateComment	S7-Graph Step name extension Step comment
S7UserTexts	用户输入的、可以输出到显示设备的文本
S7SystemTextLibrary	可集成到报文中的系统库的文本，该报文可以在运行期间动态更新，并可在PG或其它显示设备上显示
S7UserTextLibrary	可集成到报文中的系统库的文本，该报文可以在运行期间动态更新，并可在PG或其它显示设备上显示

其它可选软件包(诸如ProTool, WinCC等)的编辑器可能还有其他专用的文本类型，在此不做描述。

6.8.2 导出文件的结构

导出文件的结构如下：

$_Languages		
7(1)German(Germany)	9(1)English(USA)	
$_Type(NetworkTitle)		
要翻译的第一句	翻译	test\S7 Program(1)\Blocks\OB1
要翻译的第二句	翻译	test\S7 Program(1)\Blocks\OB1
翻译中不显示的字符	$_hide	test\S7 Program(1)\Blocks\OB1

源语言　　目标语言　　位置

1. 下面内容可能不被修改、覆盖或删除：
   - 以“$_”开始的域(这些是关键字)
   - 语言代码(在上面的解释中：9(1)是源语言英语，7(1)是目标语言德语)。
2. 每个文件包含有单个测试类型的文本。例如：文本类型是NetworkTitle ($_Type(NetworkTitle))。翻译该文件的规则包含在导出文件自身的介绍性文字中。
3. 在类型定义($_Type…)或最后一栏后必须出现关于文本或注释的其它信息。

注意：

如果目标语言栏被“512(32) $_Undefined”所覆盖，当文件导出时将不指定目标语言。为了更好地观察，你可以用目标语言替换该文本。例如“9(1)English(US)”。当导入翻译文件时，必须检验所要使用的目标语言，如果需要，选择正确的语言。

输入关键字$_hide可以隐藏而不显示目标语言。但它不会影响变量注释(Interface Comment)和符号注释(SymbolComment)。

导出文件的格式

导出文件的格式为CSV格式，当用EXCEL编辑时，必须记住只有当使用Open对话框时，才能在EXCEL正确打开CSV文件。在浏览器中双击打开CSV文件将导致文件不能使用。按下列步骤可以很方便地在Excel下使用：

1. 在EXCEL下打开导出文件
2. 将文件存储为XLS文件
3. 翻译XLS文件中的内容
4. 在EXCEL中以CSV格式保存XLS文件

注意：

导出文件可能不能更换名称。

6.8.3 管理还没有装入字体的用户文本

可以导出操作系统中还没有装入字体的用户文本，可以翻译并重新导入它们并进行保存。

但是，这些文本只能在装入字体的计算机上才能显示出来。

例如：如果用户文本要翻译成俄语，但是操作系统中没有Cyrillic字体，则需要按下列步骤进行：

1. 导出要翻译的用户文本。
2. 翻译好导出的CSV格式的文件。
3. 导入翻译好的CSV文件。结果：您的计算机中已经有了英文和俄文的项目文件。
4. 保存整个项目文件并将其发给要使用俄文的用户，如果他的机器上有Cyrillic字体，则可以显示它们。

6.8.4 日志文件

当操作多种语言文本时，将会在日志文件中生成错误记录或警报信息。该文件与所导出的文件存储在相同的目录下。

一般来讲，这些信息都带有原因。

报警

文本“xyz”在文件“xyz”中已出现，任何其他地方出现的该信息将被忽略。

解释

不管使用何种语言，文本都是被转换的基础。如果同样的一段文本被用于不同的语言，则不能被单独识别并被正确的翻译。

例如：

$ Languages	
7(1) Deutsch(Deutschland)	9(1) English (USA)
Kein	none
kenie	none
kenier	none



这些仅仅适用于标题和注释。

补救措施

将文本重新修改（例如：使用同一个德语单词而不是三个），然后重新导入。

6.8.5 优化需翻译的源文件

通过合并不同的术语和说明可以为所需翻译的文件做好准备。

例如

准备前(导出文件)：

$_Languages	
9(1)English(USA)	9(1)English(USA)
$_Type(SymbolComment)	
Auto-enab.	
Automatic enable	
Auto-enable	



合并为一个单个说明：

$_Languages	
9(1)English(USA)	9(1)English(USA)
$_Type(SymbolComment)	
Auto-enab.	Auto-enable
Automatic enable	Auto-enable
Auto-enable	Auto-enable

源语言　　　　目标语言

准备后(导入和导出后)：

$_Languages	
9(1)English(USA)	9(1)English(USA)
$_Type(SymbolComment)	
Auto-enab.	Auto-enable

源语言　　　　目标语言

6.8.6 优化翻译过程

如果现有的项目中的结构和文本与以前的项目相似，则可以优化翻译过程。

具体的，如果通过复制而创建的项目并要对其进行修改，我们推荐按下列步骤进行。

前提条件

必须有一个已经翻译好的导出文件。

步骤

1. 将该导出文件拷贝到要翻译的新的项目的文件夹中。
2. 打开新项目并导出文本(菜单命令Options > Manage Multilingual Texts > Export)。因为导出的目标文件已经存在，将会提示您是否对其覆盖。
3. 点击Add按纽。
4. 翻译导出文件(只需翻译新文本)。
5. 然后导入翻译好的文件。

6.9 微存储卡(MMC)作为一个数据载体

6.9.1 有关MMC的知识

微存储卡(MMC)是插入式存储卡，例如用于一个CPU 31xC或一个IM 151/CPU(ET 200S)。两者主要区别是高紧凑设计。

MMC采用的新的存储器概念，在下面将作简单描述。

MMC的内容

MMC可以支持装载存储器（load memory）和数据存储设备。

MMC作为装载存储器（load memory）

对于使用MMC卡的CPU，MMC卡是其全部的装载存储器。装载存储器存储了包括程序块(OB, DB, FC...)的程序以及硬件配置的内容。装载存储器的内容可以影响CPU的功能。MMC中的内容可以传送到CPU，向CPU下载的程序块将立即起作用，但是硬件配置只有在CPU复位后才起作用。

对存储器复位的响应

存储器复位后MMC中存储的块的内容依然保留。

装载和删除

可以覆盖、删除MMC中的块，但是不能恢复已经覆盖或删除的块。

访问MMC上的数据块

在MMC上，可以使用数据块和数据块的内容来处理大量的数据或用户程序很少使用的数据。新的系统操作可以用于：

- SFC 82：在装载存储器中创建数据块
- SFC 83：读取装载存储器中的数据块
- SFC 84：写装载存储器中的数据块

MMC以及口令保护

如果一个安装有MMC的CPU (例如CPU 300C系列)具有口令保护，则当在SIMATIC管理器中打开MMC时将提示您输入密码。

在STEP 7中显示存储器分配

在模板状态对话框中显示装载存储器分配时可以显示EPROM和RAM区域。在MMC卡上的块全部显示为EPROM的动作。

6.9.2 MMC作为数据载体

与STEP 7一起使用的MMC与其它任何外部数据存储器的使用方法一样。

当MMC有足够的存储空间时，可以将操作系统的文件管理器中的任何数据传送到MMC。这样，用户可以将一些图纸、指令以及功能描述方便的转交其他人员。

6.9.3 存储卡文件

存储卡文件(*.wld)用于：

- 软件PLC WinLC (WinAC Basis 和 WinAC RTX)
- SlotPLC CPU 41x-2 PCI (WinAC Slot 412 和 WinAC Slot 416)

同存储到S7-存储卡一样，WinLC或CPU 41x-2 PCI使用的程序块和系统数据可以存储到一个存储器卡文件中。这些文件中的内容对应于S7-CPU中相应存储器卡的内容。

该文件可以通过WinLC或CPU 41x-2 PCI的操作面板的菜单命令下载到其下载存储器中。

对于CPU 41x-2 PCI来说，如果CPU 41x-2 PCI没有缓冲，并且只通过一个RAM卡(自动功能)运行时，当启动PC操作系统时将自动下载该文件。

存储卡文件也是“普通的”Windows文件，可以被移动、删除或转存。

更多信息请参见WinAC产品手册。

6.9.4 在MMC上存储项目数据

用STEP 7可以在MMC上通过CPU或PG/PC存储STEP 7项目中的数据。这样在没有存储项目的时候也可以用编程器访问项目数据。

要求

当MMC插入到CPU或编程器PG/PC上时，并且已经建立了在线连接，此时只能将项目数据存储在MMC上。

确保MMC有足够的空间来满足数据存储要求。

可以存储到MMC上的数据

当MMC有足够的空间满足数据存储时，则可以向MMC传送所有的操作系统文件，数据包括：

- 全部的STEP 7项目数据
- 站组态
- 符号表

- 块和源文件
- 多语言文本
- 诸如WORD或Excel的任何其他数据

如何保存整个项目

1. 选择菜单命令**File > Memory Card File > New**。
2. 在“File name”栏中输入一个不带扩展名的文件名。
3. 在“File type”下拉表中选择文件类型“Projects (*.wld)”。
4. 点击保存按钮。

7 用不同版本STEP 7编辑项目

7.1 编辑版本 2 的项目和库

STEP 7 V5.2不再支持V2项目中的变化。当编辑V2项目或库时，将会发生不一致的情况，诸如不能用更老版本的STEP 7编辑V2项目或库。

为了能继续编辑V2项目或库，必须用V5.1以前的版本的STEP 7。

7.2 扩展用以前版本 STEP 7 创建的 DP 从站

通过导入新的*.GSD文件构成

如果在硬件目录中安装了新的设备的数据库文件(*.GSD文件)，则HW Config可以接受新的DP从站。安装后，可以在“其它现场设备”文件夹中使用它们。

如果下列条件都成立时，不能用常规方法对模块化DP从站进行重新配置和扩展：

- 从站是用以前版本的STEP 7组态的
- 在硬件目录中，从站是以类型文件表示的，而不是用*.GSD文件表示的
- 从站上已安装了一个新的*.GSD文件

解决方法

如果想在DP从站上使用新模板，该模板在*.GSD文件中被描述，则：

- 删除DP从站并重新组态。则DP从站是以*.GSD文件说明的，而不是用典型文件说明的。

如果不想使用任何一个只用*.GSD文件描述的带有新模板的从站：

- 在硬件目录窗口中的PROFIBUS-DP下，选择Other Field Devices/Compatible PROFIBUS-DP Slaves文件夹。当它们被新的*.GSD文件更新时，STEP 7将把“旧”的典型文件移到该文件夹中。在这个文件夹中，你可以发现这些之前组态用的从站，用它们可以对已组态好的DP从站进行扩展。

在STEP 7 V5.1 SP4中用GSD文件更换典型文件

对于STEP 7 V5.1 SP4，既可以更新典型文件也可以用GSD文件替换。该替换只会影响STEP 7提供的目录。

以前通过类型文件定义的DP从站的属性现在可以通过GSD文件定义，这些DP从站始终放在硬件目录中的相同的位置上。

不会删除“旧”的类型文件，而是将其移到硬件目录中的其它地方。目前它们位于目录文件夹“Other field devices\Compatible PROFIBUS DP slaves\...”中。

用STEP 7 V5.1 SP4扩展现有的DP组态

如果你要编辑用以前版本STEP 7(V5.1 SP4以前的版本)建立的项目，并要扩展一个模块化DP从站，则不能使用从硬件目录中通常的地方取出的模板或子模板。在这种情况下，使用“Other field devices\Compatible PROFIBUS DP slaves\...”中的DP从站。

用以前版本的STEP 7(V5.1 SP4以前的版本)编辑用STEP 7 SP4建立的项目

如果要组态一个用STEP 7 V5.1 SP4“更新”的DP从站，然后用以前版本的STEP 7编辑该项目，由于以前版本的STEP 7不识别所使用的GSD文件，所以不能编辑该DP从站。

解决方法：在以前版本的STEP 7中安装所需的GSD文件。在这种情况下，GSD文件存储在该项目中。如果该项目用当前使用的STEP 7进行编辑，组态时STEP 7将使用最新安装的GSD文件。

7.3 用以前版本的 STEP 7 编辑当前的组态

组态直接数据交换

在非DP主站系统中组态与DP主站的直接数据交换：

- 不能用STEP 7 V5.0 SP2(或以前的版本)
- 可以用STEP 7 V5.0 SP3和STEP 7 V5.1

如果所保存的DP主站没有将其DP主站系统进行直接数据交换参数赋值，并且试图用老版本的STEP 7 V5(STEP 7 V5.0 SP2以前的版本)对其进行编辑，将会发生：

- 将显示一个DP主站系统和从站，该从站是对所赋值的STEP 7内部数据存储区进行直接数据交换而使用的。这些DP从站不属于所显示的DP主站系统。
- 不能将一个新的DP主站系统或一个独立的DP主站连接到该DP主站上。

通过PROFIBUS DP接口在线连接到CPU

PROFIBUS DP接口没有被组态成DP主站系统：

- STEP 7 V5.0 SP2(或以前版本)：不能用该接口连接到CPU。
- STEP 7 V5.0 SP3：在编译过程中生成PROFIBUS-DP接口的系统数据；下载后可用该接口连接到CPU。

7.4 对以前版本的 SIMATIC PC 组态的添加

STEP 7 V5.1项目的PC组态(最高到SP1)

对于STEP 7 V5.1 SP2，可以用对于S7-300或S7-400站相同的方法向PC站下载通讯。然而，

通常是在存储或编译过程中生成组态文件，这样才能使用该方法向PC站传送组态信息。

该结果是“旧”的PC站不能识别包含在新生成的组态文件中的一些信息。STEP 7将自动适应这些情况：

- 如果用STEP 7 V5.1 SP2创建一个新的SIMATIC PC站，STEP 7将假设PC目标站是通过SIMATIC NET CD(7/2001)的帮助组态的，也就是假定安装了S7RTM(Runtime Manager)。用这种方法生成的组态文件可以通过“新”的PC站进行识别。
- 如果对以前版本的SIMATIC PC站的组态进行扩展(例如对用STEP 7 V5.1 SP1组态的PC站扩展)，STEP 7不会假设PC目标站是通过SIMATIC NET CD(7/2001)的帮助组态的。用这种方法生成的组态文件可以通过“旧”的PC站进行解释。

如果该缺省特性不能满足您的需要，则要按照下述内容进行修改：

在Context菜单“Configuring Hardware”中进行设定：

1. 打开PC站的硬件配置
2. 右击站窗口(白色区域)
3. 选择Context-sensitive菜单“Station Properties”
4. 检查或清除“Compatibility”选择框

在Context菜单“Configuring Networks”中进行设定：

1. 打开网络配置
2. 选择PC站
3. 选择菜单命令**Edit > Object properties**
4. 在对话框中选择“Configuration”
5. 检查或清除“Compatibility”选择框

STEP 7 V5.0项目的PC组态

如果要对用STEP 7 V5.0 SP3组态的SIMATIC PC站组态新的部件，只有SP3或更高版本才支持这些部件，此时必须对站进行转换：

1. 在SIMATIC管理器中，选中SIMATIC PC站并选择菜单命令**Edit > Object properties**。
2. 在属性对话框中的“Functions”选项中，点击“Expand”按钮对SIMATIC PC站进行转换。现在，只能用STEP 7 V5.0 SP3或更高版本对其进行编辑。

7.5 用更高版本的 STEP 7 或选件包表示模板的组态

对于STEP 7 V5.1 SP3，可以显示所有模板，即使用高版本编辑的是“老”版本STEP 7中没有的模板。也可以显示用选件包组态的模板，即使编程器上没有安装所需的相应选件包也可以显示。

在以前版本的STEP 7中，不能显示这些模板及其附属部件。在当前版本中，这些模板是可见的并可进行编辑。例如即使在其它计算机上用比较新的STEP 7建立的项目，以及由于模板具有新的特性和参数而不能用老版本的STEP 7对模板进行组态的情况下，也可以使用该功能修改用户程序。

STEP 7不识别的模板用下列图标显示：?

如果用适当的版本的STEP 7或用可兼容的选件包打开项目，将以标准的方法显示所有模板，并可进行编辑。

装有最新版本的STEP 7/选件包的PG		装有旧版本的STEP 7/无选件包的PG
	>>> --- 项目数据 --- >>>	
对最新模板表示为“识别”		对最新模板表示为“不识别”

在SIMATIC管理器中“代表（Representative）”模板的操作

在站级别下，“代表”模板表示为可见的。在该级别下的所有附属部件，诸如用户程序、系统数据和连接表均是可见的并可从SIMATIC管理器下进行下载。

也可以打开、编辑、编译和装载用户程序。但是，对于有“代表”块的项目应遵守下列限制：

- 不能复制一个包含“代表”块的站
- 不能完全使用“Save project as...”中“with reorganization”选项。在复制和重新组织项目时将丢失该模板以及该模板的所有参考和附属部件(例如用户程序)

在硬件配置中“代表（Representation）”模板的表示

在所组态的插槽上显示该模板。

可以打开该表示模板，但不能修改参数或下载。模板属性受给出的“Representative”属性的限制。不能修改站属性(例如不能增加新模版)。

可进行硬件诊断(例如在线打开一个站)，但有些限制：新的诊断操作及文本不能被识别。

在网络组态中“代表”模板的表示

也可以在NetPro上显示该“代表”模板。在这种情况下，站上的该模板名称显示为问号。

在NetPro下只能以写保护的形式打开带有“代表”模板的项目。

如果以写保护的形式打开项目，可以显示和打印网络组态。也可以得到连接状态，该状态至少包含当前使用的STEP 7版本所支持的信息。

通常情况下，不能对其修改、保存、编译和下载。

模板的安装顺序

如果模板来自更高版本的STEP 7，并已对其进行的硬件更新，可以用“真实”模板更换该“代表”模板。根据所打开的站，接收必要的硬件更新或选件包的信息，用对话方式进行安装。也可以通过菜单命令Options > Install HW Updates进行安装。

8 定义符号

8.1 绝对地址和符号地址

在STEP 7 程序中，你可以寻址I/O信号、存储位、计数器、定时器、数据块和功能块。你可以在程序中用绝对地址来访问这些地些，但是，如果使用符号地址，你的程序读起来会更容易（例如，参照你们工厂的代码系统，使用Motor_A_On或其它的标识符）。你的用户程序中的地址则可通过这些符号来寻址。

绝对地址

一个绝对地址由一个地址标识符和一个存储地址组成（如，Q4.0，I1.1，M2.0，FB21）。

符号地址

如果给绝对地址赋予符号名字，则用户程序的可读性会更好，故障诊断更容易。

STEP 7可自动地将符号地址转换成所需的绝对地址。如果要用符号名访问数组、结构、数据块、局域数据、逻辑块以及用户定义的数据类型，必须首先将符号名赋给绝对地址，然后才能对这些数据进行符号寻址。

例如，你可以将符号名MOTOR_ON赋给地址Q4.0，然后在程序指令中就可以使用MOTOR_ON寻址了。使用符号地址可以比较容易地辨别出程序中所用操作数与过程控制项目中的元素的对应关系。

注意

在符号名中不允许使用两个连续的下划线（如MOTOR__ON）。

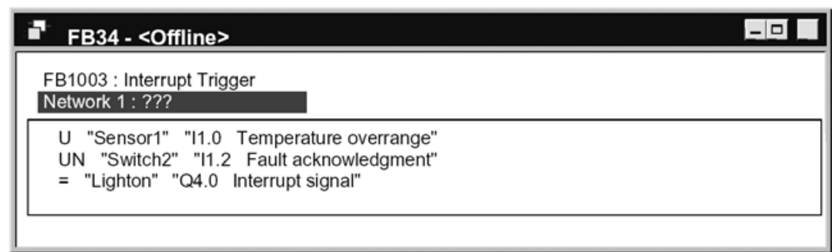
编程支持

在梯形图、功能块图及语句表这三种编程语言的表达方式中，你可以使用绝对地址或符号来输入地址、参数和块名。

使用菜单命令View>Display>Symbolic Representation，你可以在绝对地址和符号地址两种表达方式之间进行切换。

为使符号地址编程更简单，你可以显示该符号的绝对地址及注释。使用菜单命令View>Display>Symbol information，可激活此信息功能。这意味着语句表指令后面的行注释包含更多的信息。你无法在显示中进行编辑，任何修改都需在符号表或声明表中进行。

下图所示为语句表（STL）中的符号信息。



当打印一个块时，当前屏幕上带有语句注释或符号注释的表达方式会被打印出来。

8.2 共享和局域符号

符号寻址允许你用有一定含义的符号地址来替代绝对地址。将短的符号和长的注释结合起来使用，可以使编程更简单，程序文件做得更好。

你必须区分局域（块定义）符号和共享符号。

	共享符号	局域符号
有效性	• 在整个用户程序中有效， • 可以被所有的块使用， • 在所有的块中含义是一样的， • 在整个用户程序中是唯一的。	• 只在定义的块中有效 • 相同的符号可在不同的块中用于不同的目的
允许使用的字符	• 字母、数字及特殊字符， • 除0x00，0xFF及引号以外的强调号 • 如使用特殊字符，则符号须写出在引号内	• 字母 • 数字 • 下划线（_）
使用	你可以为以下各项定义共享符号： • I/O信号（I，IB，IW，ID，Q， QB，QW，QD） • I/Q输入与输出（PI，PQ） • 存储位（M，MB，MW，MD） • 定时器（T）/计数器（C） • 逻辑块（FB，FC，SFB，SFC） • 数据块（DB） • 用户定义数据类型（UDT） • 变量表（VAT）	你可以为以下各项定义局域符号： • 块参数（输入，输出及输入/输出参数） • 块的静态数据 • 块的临时数据
在哪里定义	符号表	块的变量声明表

8.3 显示共享或局域符号

如下所示，你可以在程序的指令部分区分开共享符号和局域符号。

- 符号表中定义的符号（共享）显示在引号内。
- 块变量声明表中的符号（局域）显示时前面加上“#”。

当你以LAD、FBD或STL方式输入程序时，你不必输入引号或“#”，语法会检查自动增加这些字符。

如果你担心会出现混淆，比如同一个符号在符号表和变量声明表中都使用了，那么，你必须明确地输入标志着共享符号的代码（引号）。因为在此情况下，如果没有相应的代码符号一律解释为块定义的变量（局域变量）。

如果符号地址中包含空格，则也需在此符号地址上加上共享符号的代码。

当使用STL编辑源文件时，可使用同样的特殊字符及划线。在自由编辑模式下不自动加上代码字符，但是，如果你希望避免混淆，有必要输入共享符号的代码。

注意

使用菜单命令View>Display>Symbolic Representation，你可以在所声明的符号地址和绝对地址之间进行切换。

8.4 建立地址优先权(符号地址/绝对地址)

当修改了符号表、数据块或功能块的参数名、UDT所代表的组件名或修改了多重背景时，地址优先级将帮助您修改相应的程序。

当下列状态改变时，应确保认真地设置地址优先级。为了更好地使用地址优先级，必须完成每个修改步骤，然后再开始对其它类型进行修改。

设定地址优先级，进入SIMATIC管理器，选择块文件夹然后选择菜单命令**Edit > Object Properties**。在“Address Priority”栏中进行设置。

为了对地址优先级进行优化设置，需要区分下列状态的变化：

- 对每个名称进行修正
- 修改文件名或赋值
- 新的符号、变量、参数或组件

注意：

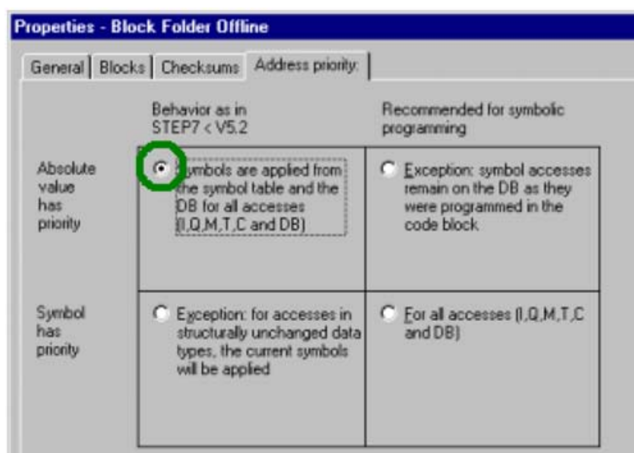
即使已经设置了符号地址优先级，在对逻辑块进行块调用时要使用绝对块号(“Call FC”或“Call FB，DB”)。

对每个名称进行修正

举例：

在符号表或程序编辑器/块编辑器中，要对名称的拼写错误进行修正。它将对符号表中的所有名称以及程序编辑器/块编辑器可以修改的参数、变量或组件的所有名称起所用。

设定地址优先级



对修改进行跟踪：

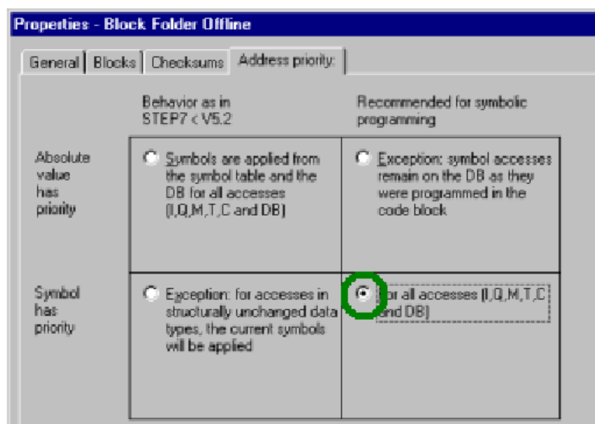
在SIMATIC管理器中，选择块文件夹并选择菜单命令Edit > Check Block Consistency。“检查块的一致性”功能可以对每个块中的必要部分进行修改。

修改文件名或赋值

举例：

- 修改符号表中已赋值的名称。
- 对符号表中已赋值的名称赋值一个新的地址。
- 在程序编辑器/块编辑器中修改变量名、参数明或组件名。

设定地址优先级



对修改进行跟踪：

在SIMATIC管理器中，选择块文件夹并选择菜单命令Edit > Check Block Consistency。“检查块的一致性”功能可以对每个块中的必要部分进行修改。

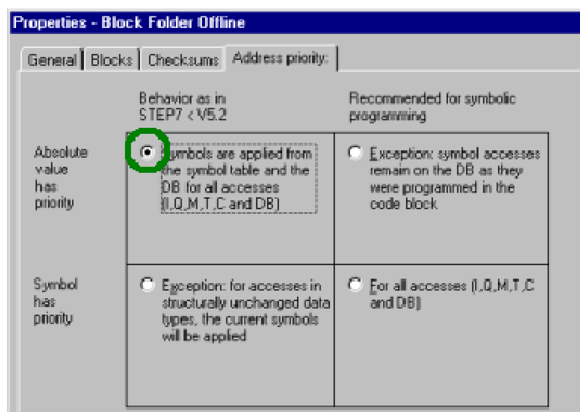
新的符号、变量、参数或组件

举例：

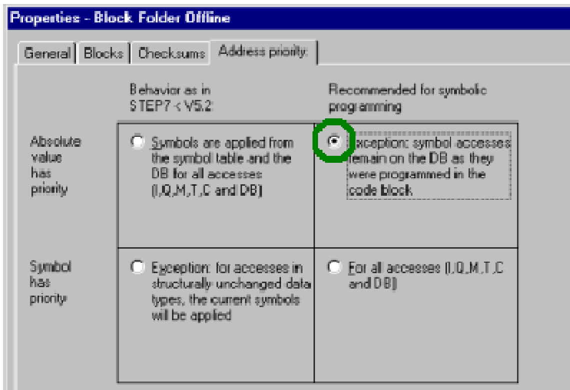
- 你正在为程序中使用的地址建立新的符号。
- 你正在为数据块、UDT或功能块添加新变量或参数。

设定地址优先级

- 在符号表中进行修改



- 在程序编辑器/块编辑器中进行修改



对修改进行跟踪：

在SIMATIC管理器中，选择块文件夹并选择菜单命令Edit > Check Block Consistency。“检查块的一致性”功能可以对每个块中的必要部分进行修改。

8.5 共享符号的符号表

共享符号在符号表中定义。

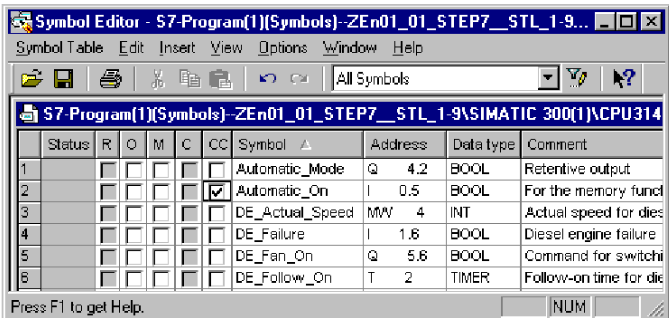
当你生成S7或M7程序时，会自动生成一个（空的）符号表（“Symbols”对象）。

有效性

符号表只对你的用户程序所连接的模板是有效的。如果你想在几个CPU中使用同样的符号，必须确保各符号表中的内容是相配的（例如，拷贝该表）。

8.5.1 符号表的结构及元素

符号表的结构



	Status	R	O	M	C	CC	Symbol	Address		Data type	Comment
1		☐	☐	☐	☐	☐	Automatic_Mode	Q	4.2	BOOL	Retentive output
2		☐	☐	☐	☐	☑	Automatic_On	I	0.5	BOOL	For the memory funct
3		☐	☐	☐	☐	☐	DE_Actual_Speed	MW	4	INT	Actual speed for dies
4		☐	☐	☐	☐	☐	DE_Failure	I	1.6	BOOL	Diesel engine failure
5		☐	☐	☐	☐	☐	DE_Fan_On	Q	5.6	BOOL	Command for switchi
6		☐	☐	☐	☐	☐	DE_Follow_On	T	2	TIMER	Follow-on time for die

行

	如果隐藏了"Special Object Properties(特殊对象属性)"栏 (没有选择菜单命令**View > Columns O,M,C,R,CC**)，则当相关行至少有一个设置了"Special Object Property"时，该行将出现该符号。

"Status"状态列

	符号名或地址与符号表中的另一个输入一致。
	符号不完整(无符号名或地址)。

R/O/M/C/CC 列

R/O/M/C/CC 列显示各符号是否赋予了特殊对象的特性：

- R（监控）意思是指使用可选软件包S7-PDIAG（V5）生成过程诊断错误定义符号。
- O指该符号能够在WinCC下被操作和监控。
- M指该符号被赋予了与符号相关的信息（SCAN）。
- C指该符号被赋予了通讯特性（只能在NCM中被选中）。
- CC指可以在符号编辑器中直接监控的符号（"Control at Contact"）。

点击检查框始能或禁止这些"特殊对象属性"。也可以通过菜单命令**Edit > Special Object Properties**进行。

Symbol（符号）列

符号名不能长于24个字符。

数据块中的地址（DBD，DBW，DBB，DBX）不能在符号表中定义。它们的名字应在数据块声明表中定义。

组织块（OB）、一些系统功能块（SFB）以及系统功能（SFC）已预先被赋予了符号名，当你为S7程序编辑符号表时可以引入这些符号名。该引入文件存在SIMATIC路径下…\S7data\Symbol\Symbol.sdf。

地址列

地址是一个特定存储区域和存储位置的缩写，例如，Input I 12.1。

输入时要对地址的语法进行检查。

"数据类型"列

在SIMATIC中可以选择多种数据类型。在数据类型区域中已有一个缺省数据类型，如果需要，用户可以修改。如果所作的修改不适合该地址或存在语法错误，当你退出该区域时会显示一条错误信息。

"注释"列

你可以给所有的符号加注释。简短的符号名与更详细的注释混合使用。可使编程更有效，使程序文件更完善。注释最长80个字符。

转换为C变量

你可以在M7程序的符号表中选中符号，然后将它们转换成相应的与ProC/C++软件选项相连接的C变量。

8.5.2 符号表中允许使用的地址和数据类型

在整个符号表中只能使用一套助记符。在Windows管理器中使用菜单命令**Options > Customize**，在“Language”选项中，可以在SIMATIC（德语）和IEC（英语）两套助记符之间进行切换。

IEC	SIMATIC	说明	数据类型	范围
I	E	输入位	BOOL	0.0 - 65535.7
IB	EB	输入字节	BYTE，CHAR	0- 65535
IW	EW	输入字	WORD，INT，S5TIME，DATE	0 - 65534
ID	ED	输入双字	DWORD，DINT，REAL， TOD，TIME	0 - 65532
Q	A	输出位	BOOL	0.0 - 65535.7
QB	AB	输出字节	BYTE，CHAR	0 - 65535
QW	AW	输出字	WORD，INT，S5TIME，DATE	0 - 65534
QD	AD	输出双字	DWORD，DINT REAL TOD，TIME	0 - 65532
M	M	存储位	BOOL	0.0 - 65535.7
MB	MB	存储字节	BYTE，CHAR	0 - 65535
MW	MW	存储字	WORD，INT，S5TIME，DATE	0 - 65534
MD	MD	存储双字	DWORD, DINT, REAL，TOD，TIME	0 - 65532
PIB	PEB	外设输入字节	BYTE，CHAR	0 - 65535
PQB	PAB	外设输出字节	BYTE，CHAR	0 - 65535
PIW	PEW	外设输入字	WORD，INT，S5TIME，DATE	0 - 65534
PQW	PAW	外设输出字	WORD，INT，S5TIME，DATE	0 - 65534
PID	PED	外设输入双字	DOWRD, DINT, REAL TOD，TIME	0 - 65532
PQD	PAD	外设输出双字	DOWRD, DINT, REAL，TOD，TIME	0 – 65532
T	T	定时器	TIMER	0 - 65535
C	Z	计数器	COUNTER	0 - 65535
FB	FB	功能块	FB	1 - 65535
OB	OB	组织块	OB	1 - 65535
DB	DB	数据块	DB，FB，SFB，UDT	0 - 65535
FC	FC	功能	FC	0 - 65535
SFB	SFB	系统功能块	SFB	0 - 65535
SFC	SFC	系统功能	SFC	0 - 65535
VAT	VAT	变量表		0 - 65535
UDT	UDT	用户定义数据类型	UDT	0 - 65535

8.5.3 符号表中不完整的和不唯一的符号

不完整的符号

你有可能存储了不完整的符号。比如，先只输入符号的名字，以后再加上相应的地址。这意味着你可能在任何时候中断符号表的编辑，暂时存储这一临时结果，另外再找时间来完成。在状态列中用 指示不完整的符号。当你要用符号来编程时（没有错误信息显示），必须事先输好符号名、地址及数据类型。

不唯一的符号是怎样出现的

当你在符号表中插入一个符号，而该符号名和/或地址已在其它行中使用，这时就会出现不唯一的符号。即：新的符号与已存在的符号不唯一。在状态列中用 指示。

上述情况会出发生在，比如你为了作一些微小的改变而拷贝并粘贴某一符号。

标定不唯一的符号

在符号表中，以图形加亮（颜色，字符）方式对不唯一的符号进行标定。这种在表达方式上的改变意味着它们仍需要编辑。你可以显示所有符号或有选择地进行显示，即只显示唯一的符号或不唯一的符号。

使符号唯一

当你修改引起符号不唯一的元素（符号和/或地址）时，不唯一的符号就变成了唯一的符号。如果两个符号相互不唯一，当改变其中一个使之成为唯一符号时，另一个也就成了唯一的符号。

8.6 输入共享符号

有三种方式进行符号输入，这些符号可以在后面的编程中使用：

- 通过对话框
  在程序输入窗口中你可以打开一个对话框，定义新的符号或对已有的符号重新定义。这种方法最好用于对单个符号的定义，比如，当你编程时发现少了一个符号或想更正某个符号。这种方法你不用显示整张符号表。
- 直接在符号表中
  你可以直接在符号表中输入符号及其绝对地址。这种方法推荐用于编辑大量符号或者为项目生成符号表的情况。因为这种方法会在屏幕上显示已经赋值的符号，你可以更容易地观察这些符号。
- 从其它表格编辑器中导入符号表
  你可以用任何一种你所喜欢的表格编辑器来生成符号数据（比如，Microsoft Excel），然后将该文件导入符号表。

8.6.1 输入符号时的注意事项

在符号表中输入新符号，将光标点中表中的第一空行并填写该单元。你可以使用菜单命令Insert>Symbol，在当前行前面插入一个新行。如果光标前面的行已包括有一个地址，则可以在通过“Address”和“Data Type”栏的预定值插入新的符号。从前一行中导出地址；缺省数据类型作为数据类型输入。

还可以使用Edit菜单中的命令拷贝并修改表中已有各项。存储然后关闭符号表，你还可以保存那些还未完全定义完的符号。

当你在表中输入符号时，须注意以下各点：

列	注　意
符号	在整个符号表中名字必须唯一。当你确认该区域的输入或退出该区域时，不唯一的符号则被标定出来。符号名最长可达24个字符。引号（″）不允许使用。
地址	当你确认或退出该区域时，程序会自动检查该地址输入是否是允许的。
数据类型	当你输入地址时，该区域被自动地赋予一个缺省数据类型。如果你修改这个缺省类型，程序会检查你的数据类型是否与地址相匹配。
注释	你可以输入注释简单地解释该符号的功能(最多80个字符)。该注释输入为可选项。

8.6.2 在对话框中输入单个共享符号

下文所描述的是如何在编辑程序块时利用对话框改变符号或定义新符号而不必显示符号表。

如果你想编辑单个符号，这种方法非常有用。如果你要编辑大量的符号，应打开符号表，直接在里面输入。

在块中激活符号显示

你可以在一个打开的块中用菜单命令View > Display > Symbolic Reprosentation，可激活符号显示。在菜单命令前有一个对号表示该表达方式激活。

输入程序时定义符号

1. 确认在块的编辑窗口中激活了符号表达方式（菜单命令View > Display > Symboilic Representation）。
2. 在程序指令中选中你想对其进行符号赋值的绝对地址。
3. 选择菜单命令Edit > Symbol。
4. 填写对话框并关闭它，用“OK”确认你的输入并且确保输入的是一个符号。

被定义的符号输入到符号表中。任何导致符号不唯一的输入都将被拒绝并显示错误信息。

在符号表中进行编辑

使用菜单命令Options > Symbol Table，可打开符号表并进行编辑。

8.6.3 在符号表中输入多个共享符号

打开符号表

有几种方法打开符号表：

- 在项目窗口双击符号表。
- 在项目窗口选中符号表，然后使用菜单命令Edit > Open Object。

处于激活状态的程序的符号表显示在自己的窗口中。此时你可以生成或编辑符号。当你在符号表生成后第一次打开时它是空的。

输入符号

在符号表中输入新符号，将光标点中表中的第一空行并填写该单元。你可以使用菜单命令Insert > Symbol，在当前行前面插入一个新的空行。还可以使用Edit菜单中的命令拷贝并修改表中已有各项。存储然后关闭符号表，你还可以保存那些还未完全定义完的符号。

符号排序

符号表中的数据可以按照符号、地址、数据类型或注释按字母顺序进行排列。

使用菜单命令View > Sort，可以打开一个对话框，重新定义分类显示以改变当前表中的排列方式。

符号筛选

你可以用筛选功能在符号表中选择某组数据。

使用菜单命令View > Filter，可以打开“Filter”（筛选）对话框。

用户可以自定义标准，只有满足标准的数据才能被在筛选后被显示出来。一般可按如下标准进行筛选：

- 符号名、地址、数据类型、注释
- 具有操作员控制及监控属性的符号、具有通讯特性的符号、信息的二进制变量的符号（存储位或过程输入）
- 有效的符号、无效的符号（不唯一、不完整）

各个标准之间通过与操作相连接。筛选后的数据以指定的字符串开始。

如果你想了解“Filter”对话框中更多的选项，可以按F1打开与之相关的在线帮助。

8.6.4 符号的大小写

在大小写字符之间无区别

以前，STEP 7中定义的符号可能与用于其它情况的符号有所不同，即区分大小写。这在STEP 7 V4.02中已经进行修改。现在不存在大小写符号的差异。

这种修改是根据客户的需要进行的，由此大大减少程序中的错误发生率。符号定义限制也支持PLCopen forum，以定义转换程序标准。

具有大小写区别的符号定义现在不再支持。在以前，例如，在符号表中可能有下述定义：

Motor1 = I 0.0

motor1 = I 1.0

这些符号的第一个字母都具有大小写区别。这种符号区别很容易造成混淆。新的符号定义避免了这种错误源。

对现有程序的影响

如果你已经使用过不同符号之间的区分标准，你可能对使用新的符号定义有困难，如果：

- 符号**只在**使用大小写字符时不同。
- 参数**只在**使用大小写字符时不同。
- 符号与参数**只在**使用大小写字符时不同。

但是，如下所述，这三种冲突都可进行分析和解决。

符号只在使用大小写字符时不同

冲突：

如果没有使用当前版本软件编辑符号表，在编译源文件时，将使用符号表中不唯一符号中的第一个符号。

如果已经编辑符号表，这些编译的符号也无效；意思是指在打开块时将不显示这些符号，并且编译包含这些符号的源文件时，将会出现错误。

排除：

打开符号表，检查符号表是否有冲突，并保存。这操作可以识别非唯一符号。然后，你可以使用“非唯一符号”滤镜显示非唯一符号，并修正。你还可以修正包含冲突的任何源文件。由于在打开块时，可以自动使用当前版本（无冲突）符号表，因此无需修改块。

参数只在使用大小写字符时不同

冲突：

在编译包含这些接口的源文件时会出现错误。可以打开使用这些接口的块，但是不能再访问这些参数中的第二个参数。如果你想存取第二个参数，在保存块后，程序将自动返回第一个参数。

排除：

为了检查包含这种冲突的块，建议使用“生成源文件”功能，生成所有程序块的源文件。如果在编译已生成源文件时出现错误，必定有冲突。

应确保参数的唯一性，以校正源文件；例如，通过“查找和替换”功能。然后，再编译文件。

只在使用大小写字符时符号与参数不同

冲突：

如果源文件中的共享符号和局部符号，只在使用在小写字符时不同，并且没有使用首字母大写，以区分共享（“符号名”）符号或局部（#符号名）符号，在编译时将总是使用局部符号。这会导致修改机器代码。

排除：

在这种情况下，建议生成所有块的新源文件。由此，可自动赋值局域和共享符号的首字母，可保证在将来编译程序时，正确处理。

8.6.5 导入/导出符号表

你可以导出当前的符号表到一个文本文件，这样就可以用任意的文本编辑器进行编辑。

你还可以将其它应用程序中生成的表导入到你的符号表中并继续编辑。这种导入功能可以用于，比如将S5程序转换为S7之后，将STEP5/ST中生成的符号赋值表导入进来。

可选择的文件格式有：*.SDF、*.ASC、*.DIF以及*.SEQ。

导出规则

可以导出整个符号表、筛选后的部分符号表或表中选中的几行。

用菜单命令**Edit>Special Object Properties**设定的符号特性不能导出。

导入规则

- 为经常使用的系统功能块（SFB）、系统功能（SFC）以及组织块（OB）预先定义的符号表已存在文件……\S7DATA\SYMBOL\SYMBOL.SDF中，需要的话，可以从该文件中导入。
- 当进行导入和导出时，符号的特性不予考虑。符号特性可通过命令菜单**Edit > Spectal Object Properties**设定。

8.6.6 导入/导出符号表的文件格式

下列文件格式可从符号表中被导入或导出：

- ASCII文件格式（ASC）
- 数据交换格式（DIF）
  可以在Microsoft Excel中打开、编辑并存储DIF文件。
- 系统数据格式（SDF）
  使用SDF文件格式从或向Microsoft Aceess应用程序导入或导出数据。
  - 可以在Microseft Access中打开、编辑并存储SDF文件。

- 在Access中，选择文件格式“Text（带分隔符）”。
- 使用双引号（”）作为文本分隔符。
- 使用逗号（，）作为单元分隔符。

- 赋值表（SEQ）
  注意：当导出符号表到一个SEQ文件时，注释长于40个字符则第40个字符以后的注释将被截去。

ASCII文件格式（ASC）

文件类型	*.ASC
结 构	数据长度，逗号分隔符，数据
示 例	126，green_phase_ped.　T　2　TIMER Duration of green phase for pedestrians 126，red_Ped.　Q　0.0　BOOL Red for pedestrians

数据交换格式（DIF）

文件类型	*.DIF
结构	一个DIF文件包含文件首部及数据

（Header）首部	表（TABLE）	DIF文件的开始
	0，1	
	“<Title>”标题	注释串
	VECTORS	文件中记录的数量
	0，<No. of records>（记录数）	
	“”	
	TUPLES	记录中的数据区域的数量
	0，<No. of cloumns>（列数）	
	“”	
	DATA	作为首部结束及数据开始的标识
	0，0	
	“”	
数据（每个记录）	<type>（类型）<numeric value>（数字值）	数据类型、数字值的标识
	<string>（字符串）	字母部分或
	V	如果没有使用字母部分

首部：　文件首部必须按规定的顺序包含TABLE、VECTORS、TUPLES及DATA。在DATA前面，DIF文件可以包含更多的可选记录类型。然而这些都被符号编辑器忽略不计。

数据：　在数据部分，每项都包含三个部分：类型的标识（数据类型）、一个数字值以及一个字母部分。

可以在Microsoft Excel中打开、编辑并存储DIF文件。不要使用重音符、变音符或其它的特殊语言字符。

系统数据格式（SDF）

文件类型	*.SDF
结构：	引号中的字符串，用逗句隔开各部分
示例：	“green_phase_ped”，“T 2”，“TIMER”，“Duration of green phase for pedestrains”， “red_ped”， “Q 0.0”，“BOOL”，“Red for pedestrains”

要在Microsoft Access中打开一个SDF文件，你应选择“Text（带分隔符）”文件格式。用双引号（”）作为文本的分隔符，用逗号（，）作为区域的分隔符。

赋值表（SEQ）

文件类型	*.SEQ
结构：	TAB 地址 TAB 符号 TAB 注释 CR
示例：	T2 green_phase_ped. Duration of green phase for pedestrains Q0.0 Red_ped. Red for Pedestrains

TAB代表制表跳格键（09H）

CR代表RETURN键（0DH）回车。

8.6.7 符号表的编辑区

对于STEP 7 V5.3，现在可以在一个符号表内选择并编辑相邻区域的符号。也就是说可以根据需要拷贝和/或截取一个变量表中的部分变量，并将其插入到另一个符号表中，或将这部分变量删除。

这样可以快速地将一个符号表中的数据传送到另一个符号表中，可以非常方便地刷新符号表。

可以选择的区域：

- 点击某行的第一栏可以快速地选择整行内容。如果你选择了从“Status”到“Comments”，也仅仅是选择了整行中的某些部分。
- 可以选择一个或多个相邻栏作为一个整个的编辑区。该区域的所有栏必须属于“Symbol”、“Address”、“Data Type”和“Comments”范围。如果进行了单个选择，则不能使用“编辑”菜单命令。
- R、O、M、C、CC栏包含有代表这些符号的特殊对象属性，只能通过“Customize”对话框(菜单命令**Options > Customize**)中的“Also copy special object properties”选项进行拷贝。

- 如果显示了R、O、M、C、CC这些列时，将对这些列中的内容进行拷贝。通过菜单命令**View > R, O, M, C, CC Columns**可以显示或隐藏这些列。

编辑一个符号表的步骤：

1. 使用下述方法选择要在符号表中进行编辑的区域：
   - 使用鼠标，点击起始单元，同时按住鼠标左键，将光标移动到要选择的地方。
   - 使用键盘，按住SHIFT键及光标键选择编辑区域。
2. 所选择的区域以反白显示。选择的第一个单元以正常显示，并有一个边框。
3. 根据需要对所选择的区域进行编辑。

9 程序块和程序库的生成

9.1 选择一种编辑方法

根据生成程序时所选用的编程语言，我们可以用增量输入方式编程或用自由编辑（文本）方式编程。

增量编辑器适用于梯形逻辑图、功能块图、语句表以及S7-GRAPH等编程语言

增量输入方式编辑器适用于梯形图、FBD、STL和S7-GRAPH，用该方式生成的块存放在用户程序中。如果你想立刻检查所输入的内容，就应该用增量输入方式。这种编程方式尤其适合于初学者。用增量输入方式输入时，每一行或每个元素都会立即进行句法检查。只有改正了所指出的错误才能完成输入。句法修正过的所有输入经过自动编译存到用户程序中。

任何用于语句中的符号必须事先定义。如果在程序块中使用了没有定义的符号，则该块不能完全编译，但是这种不一致的临时版程序可以存盘。

源代码（文本）编辑器适用于语句表、S7 SCL、S7 HiGraph编程语言

在源代码编辑器中，是用**源文件**的形式生成用户程序，该文件再编译成各类程序块。

我们建议您使用源代码进行编辑，因为它是高效地进行编程和监视。

在文本文件中编辑程序或块的源代码，然后再进行编译。

文本文件（源文件）存在你的项目中S7 Program下的“source file”文件夹中，例如：**STL source file**或**SCL source file**。一个源文件可包含一个块或多个块的程序代码。用文本编辑器进行STL和SCL编程可生成**OB**、**FB**、**FC**、**DB** 及**UDT**（用户定义数据类型）的代码，也可以生成整个用户程序。CPU的所有程序（意味着所有的块）可包含在一个文本文件中。

当你编译源文件时，就会生成各种类型的块，并在用户程序中存起来。在文件中使用的任何符号必须在编译之前加以定义。在编译的过程中，由编译器报告错误。

为了顺利地通过编译，在编程语言中使用专门的句法是非常重要的。只有当选择了相容性检查命令或将源文件编译成程序块时，才运行句法检查功能。

9.2 选择编程语言

为编辑器设置编程语言

当用户要生成某程序块或源文件时，应在对象的属性中设置用于生成该块或源文件的编程语言和编辑器类型。该输入确定当该程序块或源文件打开时，启动的是哪种编辑器。

启动编辑器

在SIMATIC管理器中，用双击相应的对象（块、源文件，等），或选择菜单命令**Edit > Open Object**，或在工具条中选择相应的按钮，来启动相应的语言的编辑器。

在表中列出的编程语言都可用于生成S7程序。在标准的STEP 7软件包中包括LAD、FBD、STL。也可购买做为可选软件包的其它的编程语言。

你可以选择一系列不同的编程方法（梯形逻辑、功能块图、语句表、高级语言、顺序控制或状态图形）。还可以选择是用文本方式编程，还是用图形方式编程。

选择好编程语言，也就确定了可以用哪种输入方式（•）。

编程语言	用户类	应用	增量输入	自由编辑方式	可从CPU备份程序块
语句表STL	愿意用类似于机器码语言编程的用户	程序在运行时间和存贮空间要求上最优	•	•	•
梯形逻辑LAD	习惯电路图的用户	编写逻辑控制程序	•	—	•
功能块图FBD	熟悉布尔代数逻辑图的用户	编写逻辑控制程序	•	—	•
F-LAD，F-FBD 可选软件包	熟悉LAD和FBD的用户	编写F系统的程序	•	—	•
SCL（结构控制语言）可选软件包	用高级语言，如PASCAL或C编程的用户	数据处理任务程序	—	•	—
S7-GRAPH 可选软件包	有技术背景，没有PLC编程经验的用户	以顺序过程的描述很方便	•	—	•
HiGraph 可选软件包	有技术背景，没有PLC编程经验的用户	对异步非顺序过程的描述很方便	—	•	—
CFC可选软件包	有技术背景，没有PLC编程经验的用户	适于连续过程的描述	—	—	—

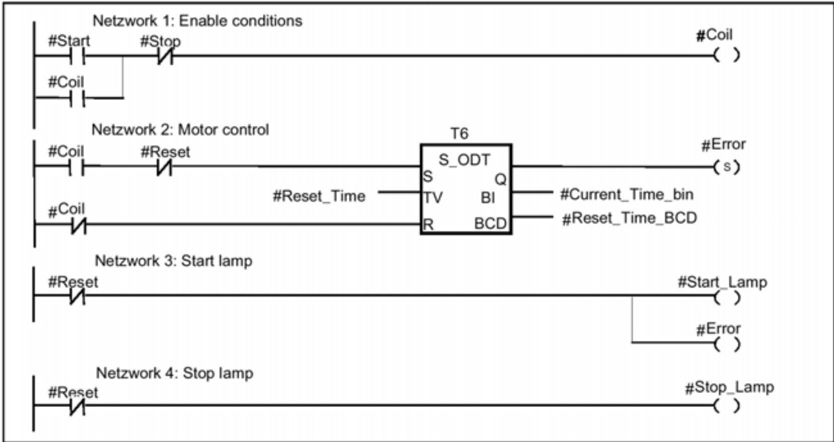
如果程序块中没有错误，可将其在梯形逻辑、功能块图和语句表之间进行切换。如果有部分程序不能切换，则用语句表显示。

可用源文件的语句表生成各程序块，也可将各程序块反编译到源文件中。

9.2.1 梯形逻辑编程语言（LAD）

图形编程语言梯形逻辑是基于电路图表示法的基础之上，在程序段中将电路图中的元素如常开触点和常闭触点组合而成。一个逻辑块的程序部分由一段或多段程序组成。

梯形逻辑程序段举例



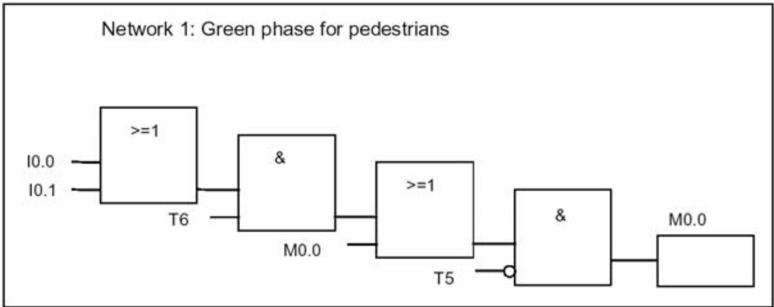
梯形逻辑编程语言包含在STEP 7标准软件包中。梯形逻辑程序是用增量编辑器生成。

9.2.2 功能块图编程语言（FBD）

编程语言功能块图（FBD）使用类似于布尔代数的图形逻辑符号来表示控制逻辑。一些复杂功能诸如算术功能等，可直接用逻辑框表示。

FBD编程语言包含在STEP 7标准软件包中。

举例：FBD中的一个程序段



在FBD方法中用增量编辑器生成程序

9.2.3 语句表编程语言（STL）

编程语言的另一种表示法是语句表，它类似于机器码的一种文本语言。每条语句对应CPU处理程序中的一步。多条语句可组成一程序段。

举例：语句表中的程序段

```
        Network 1: Control drain valve
A (
O
O   #Coil
)
AN  #Close
=   #Coil
        Network 2: Display "Valve open"
A   #Coil
=   #Disp_open
        Network 3: Display "Valve closed"
AN  #Coil
=   #Disp_closed
```

语句表编程语言类型包含在STEP 7标准软件包中。用这种语言，你可以用增量编辑器编辑S7块，在源代码编辑器中可以创建和编译STL程序源文件，以生成程序块。

9.2.4 S7 SCL编程语言

编程语言SCL（结构化控制语言）是一个可选软件包，它是按照国际电工技术委员会IEC1131-3标准定义的高级的文本语言。它类似与PASCAL类型语言，在编写诸如回路和条件分支时，用其高级语言指令要比STL容易。因此，SCL适合于公式计算，复杂的最优化算法或管理大量的数据。

S7 SCL程序是在源代码编辑器中编写的。

例如：

```
FUNCTION_BLOCK FB20
VAR_INPUT
ENDVAL：                INT；
END_VAR
VAR_IN_OUT
IQ1：                   REAL；
END_VAR
VAR
INDEX：                 INT；
END_VAR
```

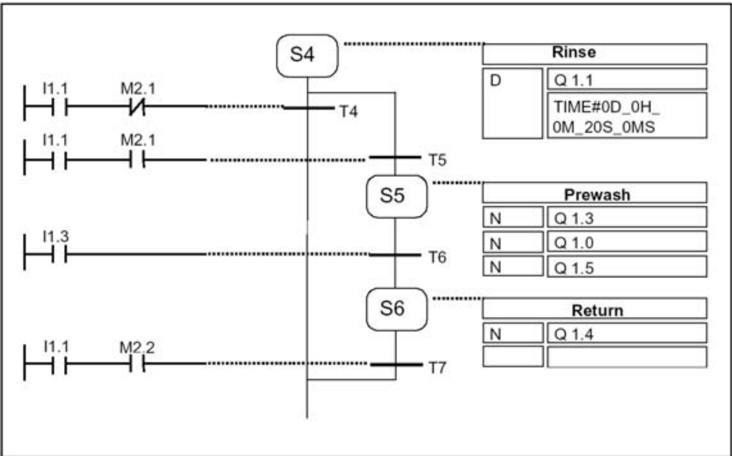
```
BEGIN
CONTROL：=FALSE；
FOR INDEX：= 1 TO ENDVALUE DO
        IQ1：= IQ1 * 2；
        IF IQ1 >10000 THEN
        CONTROL = TRUE
        END_IF
END_FOR；
END_FUNCTION_BLOCK
```

9.2.5 S7-GRAPH 编程语言（顺序控制）

图形编程语言 S7-GRAPH属于可选软件包，适用于顺序控制的编程。它包括生成一系列顺序步，确定每一步的内容，以及步与步之间的转换条件。编写每一步的程序要用特殊的编程语言（类似于语句表），转换条件是在梯形逻辑编程器中输入（梯形逻辑语言的流线型版本）。

S7-GRAPH 表达复杂的顺序控制非常清晰，用于编程及故障诊断更为有效。

S7-GRAPH的顺序控制程序举例



程序块的生成

用 S7-GRAPH 编辑器，将生成含有顺控器的功能块程序。相应的背景数据块中含有顺控器的数据，例如：FB的参数，顺序步和转换条件。用S7-GRAPH编辑器能自动生成背景数据块。

源文件

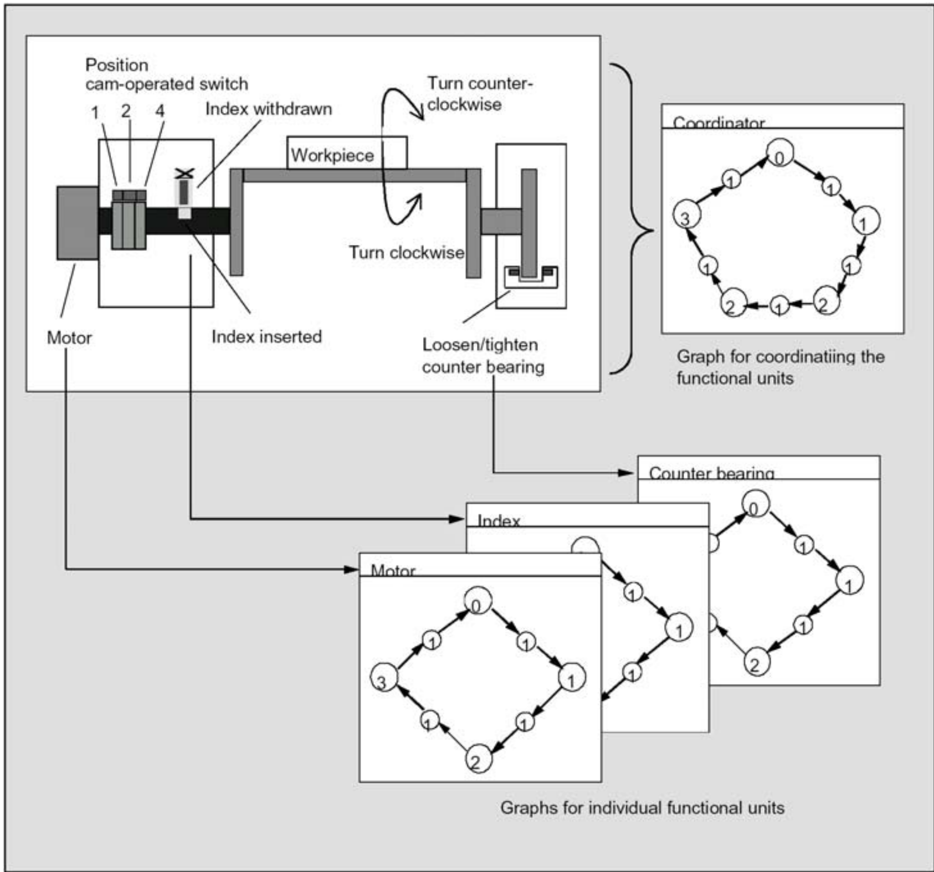
通过S7-GRAPH 生成的功能块可以产生一个文本源文件（图形源文件），该源文件可由操作员面板（OP）或操作员接口文本显示（TD）编译显示成顺控器。

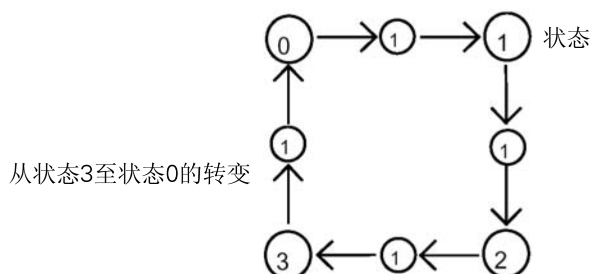
9.2.6 S7 HiGraph编程语言（状态图形）

图形编程语言 S7 HiGraph属于可选软件包，可以将程序中的各块做为状态图形编程。这种方法将你的项目分成不同的功能单元，每个单元有不同的状态。不同状态之间的切换要定义转换条件。用类似于语句表的放大型语言描述赋给状态的功能以及状态之间转换的条件。

每个功能单元都用一个图形来描述该单元的特性。整个项目的各个图形组合起来为图形组。各功能单元的同步信息可在图形之间交换。

各功能单元的状态条件的清晰表示，使得系统编程成为可能，故障诊断简单易行。与S7 Graph不同，在 S7 HiGraph中任何时候只能一个状态（在S7 Graph中：“步”）是激活的。下列图形为功能单元的图形是怎样生成的（举例）。





图形组存在HiGraph源文件中 S7 program之下的“Source”文件夹中。该源文件可编译成用户程序中的S7程序块。

句法和形式参数在图形最后输入时检查（当工作窗口关闭时）。地址和符号在源文件编译时检查。

9.2.7 S7 CFC 编程语言

可选软件包CFC（*Continuous Function Chart，连续功能图*），是一种用图形的方法连接复杂功能的编程语言。

编程语言S7 CFC用于连接已存在的各种功能。有许多标准功能不需要用户编程，而是可以使用含有标准块（例如：逻辑、算术、控制和数据处理等功能）的程序库。使用CFC不需要用户掌握详细的编程知识以及有关可编程序控制方面的专门知识。只需要具有行业所必需的工艺技术方面的知识就可以。

用户生成的程序块可按自己的意愿进行连接，连接的方法分不同的情况，如果用SIMATIC S7，可用S7编程语言中的任一种，如果是用于SIMATIC M7则用C/C++编程语言。

程序是按CFC图表生成并存贮。这些程序存在S7 program下面的“Charts”文件夹中。这些图表可编译成用户程序中的S7程序块。

9.3 生成软件块

9.3.1 块文件夹

用户可以按下列形式生成S7 CPU 中的程序：

- 块
- 源文件

文件夹“Blocks（块）”在S7 program路径之下，用于存放各程序块。

该文件夹中的程序块需要用户下载到S7 CPU 中用于执行自动控制任务。这些可装载的程序块包括逻辑块（OB，FB，FC）和数据块（DB）块文件夹中会自动生成一个空的组织块OB1，在S7 CPU中必须用该组织块来执行用户程序。

块文件夹还包含下列对象：

- 用户生成的用户定义数据类型（UDT）。UDT可使编程变得容易，但不需要下载到CPU中。
- 用户可生成变量表（VAT）在测试用户程序时用于监视和修改变量。变量表不下载到CPU中。
- 在对象“系统数据”（系统数据块）中含有系统信息（系统组态，系统参数）。这些系统数据块是当用户进行硬件组态时提供数据并生成的。
- 系统功能（SFC）和系统功能块（SFB），需要时可在用户程序中调用。但是，用户不能自己编写SFC和SFB。

除了系统数据块（只能通过对可编程序控制器的进行组态编辑和生成）以外，用户程序中的其它块都需要用相应的编辑器进行编辑。其编辑器在双击相应的块时自动打开。

注意

用源文件编写的各程序块，当编译后也存在块文件夹之下。

9.3.2 用户定义的数据类型（UDT）

用户定义数据类型是一种特殊的数据结构，由用户自己生成，该结构一旦定义好了之后，可在整个S7程序中使用。

- 用户定义数据类型可在逻辑块（FC，FB，OB）变量表中用做数据类型元素或复杂的数据类型。或者在数据块（DB）中做为数据的变量类型使用。其优点是，用户只需定义一次某种常用的特殊的数据结构，然后，可多次使用。
- 用户定义数据类型也可做为样板，用于生成与其数据结构相同的数据块。这就意味着，用户只需生成一次数据结构，然后，简单地将用户定义数据类型分配给所要生成的数据块。（例如配方：数据块的结构总是相同的，只是数值不同。）

用户定义数据类型在SIMATIC管理器中生成，或在增量编辑器中生成——与其它类型块相同。

用户定义数据类型（UDT）的结构

当你打开一个用户定义数据类型时，出现一个新的工作窗口，在该窗口中用变量表的形式显示用户定义数据类型。

- 在第一行和最后一行已经有STRUCT和END_STRUCT用于表明用户定义数据类型的开始和结束。用户不能编辑这两行。
- 用户在变量表的第二行相应的列中开始输入用户定义数据类型。

- 用户可从下列数据类型中建立用户定义数据类型：
  - 基本数据类型
  - 复杂数据类型
  - 已存在的用户定义数据类型

S7用户程序中的用户定义数据类型不下载到S7 CPU中。既可用增量输入编辑器直接生成和编辑，也可通过源文件编译时生成。

9.3.3 块属性

块属性可使用户更容易辨识所生成的各程序块，并且还可以对这些程序块加以保护，防止非法修改。

用户应在某程序块建立时编辑块属性。除了用户编辑的属性，属性对话框也显示信息：但是用户不能编辑这些信息。

块属性和系统属性也在SIMATIC管理器中块的对象属性中显示。在此用户只能编辑名称，系列，作者和版本等属性。

当通过SIMATIC管理器建一个新块时，用户可编辑对象属性。如果不是用SIMATIC管理器，而是用其它编辑器生成程序块，则该块的某些属性（如：编程语言）会自动地存入对象属性中。

注意

用于S7程序块编程的记忆系统，可通过SIMATIC 管理器菜单命令**Options > Customize** 和“Language”表设置。

块属性表

当输入块属性时，输入顺序如下表。

关键字/属性	含 义	举 例
[KNOW_HOW_PROTECT]	块保护；使用该指令后，当该块进行编译时，使程序部分不可见。块的界面可以显示，但不能改变。	KNOW_HOW_PROTECT
[AUTHOR:]	作者名，公司名，部门名或其它名你（最多为没有空格的8个字符）	AUTHOR: Siemens, 没有关键字
[FAMILY:]	块系列名：例如：控制器（最多为没有空格的8个字符）	FAMILY: Controllers 没有关键字。
[NAME:]	块名（最多8个字符）	NAME: PID 没有关键字。
[VERSION:int1,int2]	块的版本号（两个数的范围均在0到15之间，意为0.0到15.15）	VERSION: 3.10
[CODE_VERSION1]	功能块能否有多重背景的标识符。如果想用多重背景则功能块不应有该属性	CODE_VERSION1

关键字/属性	含　义	举　例
[UNLINKED]只适用于DB	具有UNLINKED属性的数据块只能存储在装载存储器中。他们不占用工作存储器空间，也不与程序链接。他们不能用MC7命令进行访问。只能通过SFC 20 BLKMOV (S7-300/400)或SFC 83 READ_DBL (S7-300C)命令残能将该DB的内容传送到工作存储器。	
[Non-Retain]	每次电源上电以及CPU每次从STOP转换到RUN时，具有该属性的数据块将对装载值复位。	
[READ_ONLY]只适用于DB	数据块的写保护；其数据只能读，不能修改	READ_ONLY

块保护命令KNOW_HOW_PROTECT 具有下列效果：

- 如果想在增量STL、FBD或梯形图编辑器中输出编译后的保护块，则该块的程序部分不能显示。
- 该块的参数声明表中只显示类型为 var_in、var_out和var_in_out的变量。类型为var_stat和var_temp的变量隐含。

对应：块属性与块类型

下表所示为哪种块类型可具有哪些块属性：

特　性	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	•	•	•	•	—
AUTHOR	•	•	•	•	—
FAMILY	•	•	•	•	—
NAME	•	•	•	•	—
VERSION	•	•	•	•	—
UNLINKED	—	—	—	•	—
READ_ONLY	—	—	—	•	—
Non-Retain	—	—	—	•	—

当你用源文件编写程序块时，可设置KNOW_HOW_PROTECT属性，并在“块属性”对话框中显示，但不能修改。

9.3.4 显示块长度

块长度的显示单位为“bytes（字节）”

块文件夹属性显示

下述长度将显示在离线窗口中的块文件夹属性中。

- 可编程控制器的装入存储器大小（不包括系统数据的所有块之和）。
- 可编程控制器的工作存储器大小（不包括系统数据的所有块之和）。
- 编程器（PG/PC）上的块长度不显示在块文件夹属性中。

块属性显示

在块属性中将显示以下内容：

- 所需局域数据数量：局域数据按字节表示的大小
- MC7：MC7按字节表示大小或DB用户数据的大小
- 编程控制器中输入存储器的大小
- 编程控制器中工作存储器的大小只有在识别出硬件分配时，才显示

所有显示信息，与块是位于在线窗口还是离线窗口中无关。

SIMATIC管理器中显示（详细视图）

如果打开一个块文件夹，并选择了“Details View（详细视图）”，工作存储器的要求将显示在项目窗口中，与文件夹是位于在线窗口中还是离线窗口中无关。

选择所有有关块，可以计算块长度的总和。在这种情况下，所选块长度的总和将显示在SIMATIC Manager的状态栏中。

对于没有下载到可编程控制器中的块，将不显示其长度（例如变量表）。

编程器（PG/PC）上的块长度不显示在详细视图中。

9.3.5 块比较

介绍

可以用以下两种方法之一，可以比较位于不同位置的块：

- 在SIMATIC管理器中选择菜单命令**Options > Compare Blocks**。在出现的“Compare Blocks - Results”对话框中点击“Go to”按钮。在程序编辑器的“Comparison”栏中显示比较结果。
- 进入程序编辑器，选择**Options > Compare On-/Offline Partners**菜单命令。

在下面几节中将解释如何进行块比较功能。逻辑块(OB、FB、FC)和数据块(DB)在比较时是有区别的。

如何进行块比较：逻辑块

第一步，STEP 7将对逻辑块接口的时间标记进行比较。如果时间标记相同，则STEP 7假设其接口也是相同的。

如果时间标记不同，STEP 7将分“节”一步一步对接口的数据类型进行比较。当发现不同时，STEP 7将对第一个不同内容进行指示。比较时还将对多重背景和UDT进行比较。如果各部分数据类型相同，STEP 7将比较各个变量的初始值。比较过程中，将显示所有差别。

第二步，STEP 7将按网络段顺序检查程序代码 (如果没有选择“Execute code comparison”(执行代码比较)选项，如果点击了程序编辑器中的“Go to”按钮，则将始终比较程序代码)。

首先将检测插入的和删除的网络段。比较结果将显示在一个块的网络段上。

其次比较剩余的网络段，直到发现第一条差别。语句将按以下方式比较：

- 设置“Absolute address has priority(绝对地址优先)”时将按绝对地址比较
- 设置“Symbol has priority(符号优先)”时将按符号比较

如果操作指令和地址是相同的，则认为语句一致。

如果所比较的块是以不同的编程语言编写的，则STEP 7将基于STL语言对其进行比较。

离线-离线比较的特性：

与离线-在线比较相比，在离线-离线比较时，STEP 7也检查变量名是否不同。因为在线时变量名替换为符号名，所以在离线-离线比较时不能进行该步比较。语句的注释以及其他一些块的属性（比如S7-PDIAG的信息）也不会被比较。

如何进行块比较：数据块

第一步，STEP 7将对数据块接口的时间标记进行比较(与逻辑块比较相同)。如果时间标记相同，则STEP 7假设其数据结构也是相同的。

如果接口时间标记不同，STEP 7将比较数据结构，直到发现第一个不同为止。如果各“节”的数据结构相同，STEP 7将比较初始值和当前值。比较过程中，所有差别都将被显示。

离线-离线比较的特性：

与离线-在线比较相比，在离线-离线比较时，STEP 7也检查变量名是否不同。因为在线时变量名替换为符号名，所以在离线-离线比较时不能进行该步比较。

比较时不包括数据块中UDT的注释和结构。

如何进行数据类型(UDT)比较

第一步，STEP 7将对数据类型的时间标记进行比较(与数据块比较相同)。如果时间标记相同，则STEP 7假设其数据结构也是相同的。

如果接口时间标记不同，STEP 7将比较数据结构，直到发现第一个不同为止。如果各节的数据结构相同，STEP 7将比较初始值。比较过程中，所有差别都将被显示。

如何在程序编辑器内进行比较

1. 打开要比较的块。
2. 选择菜单命令Options > Compare On-/Offline Partners。

   如果能在线访问，则比较结果将显示在程序编辑器窗口的下方。

   提示：如果两个网络段不同，则可以在相应行上双击便可打开相应的网络段。

如何在SIMATIC Manager内进行比较

1. 在SIMATIC Manager中选择要比较的块文件夹或要比较的块。
2. 选择菜单命令**Options > Compare Blocks**。
3. 在显示的“Compare Blocks(块比较)”对话框中选择比较类型(ONLINE/offline or Path1/Path2)。

4. 对于Path1/Path2比较：在SIMATIC Manager中选择要比较的块文件夹或要比较的块。这些块将自动地输入到对话框中。
5. 如果要比较SDB，选择“Including SDB”选择框。
6. 如果要比较程序代码，选择“Execute code comparison”选择框。在详细比较中，除了比较程序块中与执行相关的部分外(接口和程序代码)，也显示任何局部变量名和参数的变化。此外，还可以选择“Including blocks created in different programming languages(包括由不同编程语言生成的块)”选择框来比较由不同编程语言生成的块。此时的比较时基于STL语言进行的。
7. 在对话框中点击“OK”确认设置。

   比较结果显示在“Compare Blocks – Results(比较块–结果)”的对话框中。

   点击对话框中的“Details”按钮可以显示所比较块的属性(例如上次修改时间等)。

   打开程序编辑器，在窗口的下面将显示比较结果，点击“Go to”按钮。

注意：

当离线块文件夹与在线块文件夹进行比较时，只比较下载的块(OB、FB…)。

当比较offline/online或Path1/Path2时，即使一些块没有装载，也会对所有块进行比较，例如变量表或UDT。

9.3.6 再接线

下列块和地址可以被再接线：

- 输入、输出
- 存储位定时器计数器
- 功能、功能块

要进行再接线：

1. 在SIMATIC管理器中，选择含有要再接线的单个块的“Blocks”文件夹。
2. 选择菜单命令**Options＞Rewire**。
3. 在“再接线”对话框中的表中输入所要的替代（旧地址/新地址）。
4. 如果你想要的再接线的地址区域（字节、字、双字），选择选项“All addresses within the specified address area（指定地址区域内的所有地址）。”

   例如：你输入IW0和IW4作为地址区域。则地址I0.0-I1.7被再接线为I4.0-I5.7。要进行再接线的区域的地址（如，I0.1）不能够在表中再单独输入。
5. 点击“OK”按钮。

这就启动了再接线过程。在再接线完成后，你可以在一个对话框中指定是否要看关于再接

线的信息文件。该信息文件包含“旧地址”和“新地址”的列表。对每个块还列出了对其所作的接线处理的个数。

在再接线时，应注意以下事项：

- 再接线块时（即重新命名），不能存在新块。如果存在，过程将中断。
- 当再接线功能块（FB）时，实例数据块将自动赋值给再接线FB。实例DB不能改变，即，仍保持原有的DB号。

9.3.7 块及参数属性

有关特性的描述可查找在线帮助中的系统特性。

9.4 有关程序库

程序库用于存放SIMATIC S7/M7中可多次使用的程序部件。这些程序部件可从已有的项目中复制到程序库中，也可以直接在程序库中生成。该程序库与其它项目无关。

如果在S7 program下的程序库中存放有用户希望多次调用的块，可节省大量的编程时间并提高效率。可以将这些程序块拷贝到用户程序中所需要的地方。

在程序库里生成S7/M7程序，与在项目中的做法相同，只是没有测试功能。

生成程序库

生成程序库的方法与生成项目的方法一样，用菜单命令 **File > New**。新生成的程序库，其目录在菜单命令**Option>Customize**中的“General”页设置。

注意

SIMATIC管理器允许名字多于八个字符。但是，程序库目录名多于8个字符将截去。所以各程序库的名称在前8个字符中不能相同。名字不必区分大小写。当该目录在浏览器中打开时，可见全名。但当浏览该目录时，则只出现短名。注意，不能在老版本的STEP 7项目中，使用新版STEP 7程序库中的程序块。

打开程序库

用菜单命令File > Open打开一个已存在的程序库。然后选择对话框中的程序库名。最后打开程序库窗口。

注意

如果在程序库表中找不到所需的程序库，则可用“打开”对话框中的“Browse”浏览键。在标准的浏览器窗口中通过显示的目录结构来查找所需的程序库。

注意，文件名总是对应程序库生成时的原始名。在SIMATIC管理器中修改文件名不改变文件级的文件名。

当你选择了程序库，则将其列入程序库表中。用户可用菜单命令File > Manage，修改程序库表中的输入。

复制程序库

用户可用菜单命令File > Save as，将一个程序库存在另一个名下来复制程序库。

使用菜单命令Edit > Copy，可复制程序库中的某一部分，如：程序、块、源文件等。

删除程序库

使用菜单命令File > Delete，可删除一个程序库。

用菜单命令Edit > Delete，可删除程序库中的某一部分，如：程序、块、源文件等。

9.4.1 程序库的等级结构

程序库结构按等级进行，与项目一样：

- 程序库可含有S7/M7 programs。
- 一个S7 program可含有一个“块（Blocks）”文件夹（用户程序）一个“源文件（Source Files）”文件夹，一个“图表（Charts）”文件夹，和一个“符号（Symbols）”对象（符号表）。
- 一个M7 program可含有用于可编程M7模板的图表和C程序以及一个“Symbols（符号）”对象（符号表）和一个“Blocks（块）”文件夹用于数据块和变量表。
- “Blocks（块）”文件夹包含有可下载到S7 CPU中的各种程序块。在文件夹中的变量表（VAT）和用户定义数据类型不能下载到CPU中。
- “Source Files（源文件）”文件夹包括各种编程语言生成的源文件程序。
- “Chart（图表）”文件夹包含CFC图表（只有安装了CFC可选软件后才出现）。

当用户插入一个新的S7/M7 program时，“Blocks”文件夹、“Source Files”文件夹（只有S7）和“Symbols”对象会自动插入。

9.4.2 标准库总览

在STEP 7标准软件包中包括标准程序库：

- **系统功能块**：系统功能块（SFB）和系统功能（SFC）
- **S5-S7转换块**：用于转换STEP 5程序的块
- **IEC功能块**：用于IEC功能的块，诸如：处理时间和日期信息、比较操作、字符串处理以及选择最大值和最小值
- **组织块**：标准组织块（OB）
- **PID控制块**：用于PID控制的功能块（FB）
- **通讯块**：SIMATIC NET CP使用的功能（FC）和功能块
- **TI-S7转换块**：通常使用的标准功能。
- **其它块**：用于时间标签和TOD同步的块

当安装可选软件包时，可增加其它库

删除及安装所提供的程序库

在SIMATIC管理器中，用户可以删除所提供的程序库，然后再重新安装。要想安装程序库，用户必须重新执行一次STEP 7的安装程序。

注意

当用户安装STEP 7时，所提供的程序库总是自动安装。如果用户编辑了这些程序库，在重新安装STEP 7时，修改过的程序库将会被原始的程序库覆盖。

由于这个原因，用户应在做任何改动之前，复制所提供的程序库，然后只在复制的程序库中进行编辑。

10　逻辑块的生成

10.1　生成逻辑块的基础

10.1.1　程序编辑器窗口的结构

程序编辑器的窗口分为以下几个区域：

表格

“Program Elements”标签中显示可以插入到LAD、FBD或STL程序的程序单元表。“Call Structure”标签显示当前S7程序中块调用的层次。

变量声明表

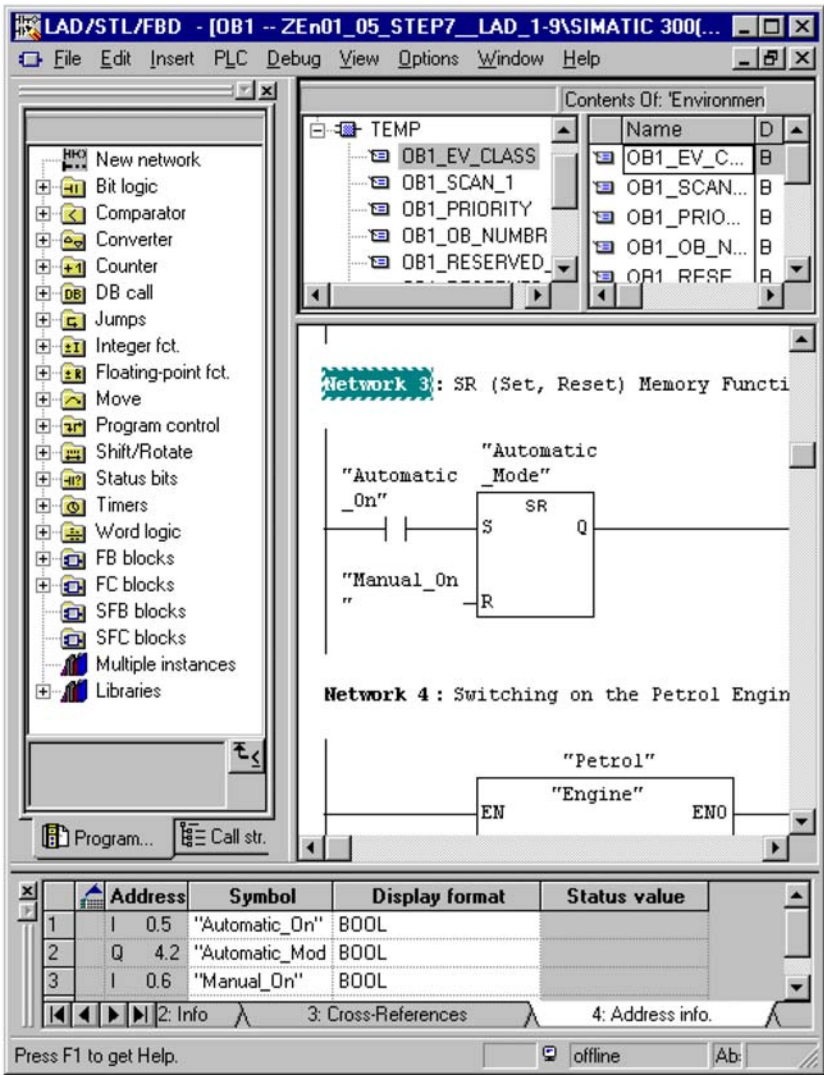
变量声明表分为“Variable Table”(变量表)和“Variable Detail View”(变量详细显示)两个部分。

指令

指令显示PLC要处理的块代码。它包含一个或多个编程网络。

详细

“Details”窗口中的各种标签提供诸如显示故障信息、编辑符号、提供地址信息、控制地址、块比较和硬件诊断故障等功能。



10.1.2 生成逻辑块

逻辑块（OB、FB、FC）具有变量声明表部分、程序指令部分和属性部分。当用户编程时，必须编辑下列三部分：

- **变量声明表**：在变量声明表中，用户可以规定参数及参数的系统特性和本地块定义的变量。
- **程序指令部分**：在程序指令部分，用户编写能被可编程序控制器执行的指令代码。这些程序可分为一段或多段，可以使用编程语言梯形逻辑（LAD）、功能块图（FBD）或语句表（STL）来生成程序。
- **块属性**：块属性中包含附加的信息，如：由系统输入的时间标签或路径。此外，用户可输入自己的内容，如：块名、系列名、版本号和作者，并且用户可将系统属性分配给程序块。

原则上，用户编辑逻辑块各部分的顺序并不重要，各部分可随时修改及增加。



注意：

如果用户希望使用符号表中的符号，必须首先检查它们是否完整及进行必要的修改。

10.1.3 LAD/STL/FBD程序编辑器的缺省设置

在开始编程之前，用户应熟悉编辑器的设置，这样在编程时感觉容易并且方便。

使用菜单命令Options > Customize，打开一个标签对话框。在不同的“Editor（编辑器）”标签中，用户可为逻辑块编程做以下预置，例如“逻辑器”页中：

- 文本和表格中的字体（类型和大小）。
- 在一个新块中是否显示符号和文字注释。

用户在编辑状态中，用View >...下拉式菜单中的命令可以修改编程语言表示法、文字注释和符号的设置。

在“LAD/FBD”页中，用户可以改变主要部分的颜色，例如：程序段或语句行。

10.1.4 逻辑块和源文件的访问授权

当编辑一个项目时，通常使用一个共同的数据库，这就意味着项目组的全体成员也许想在同时访问同一个块或数据源。

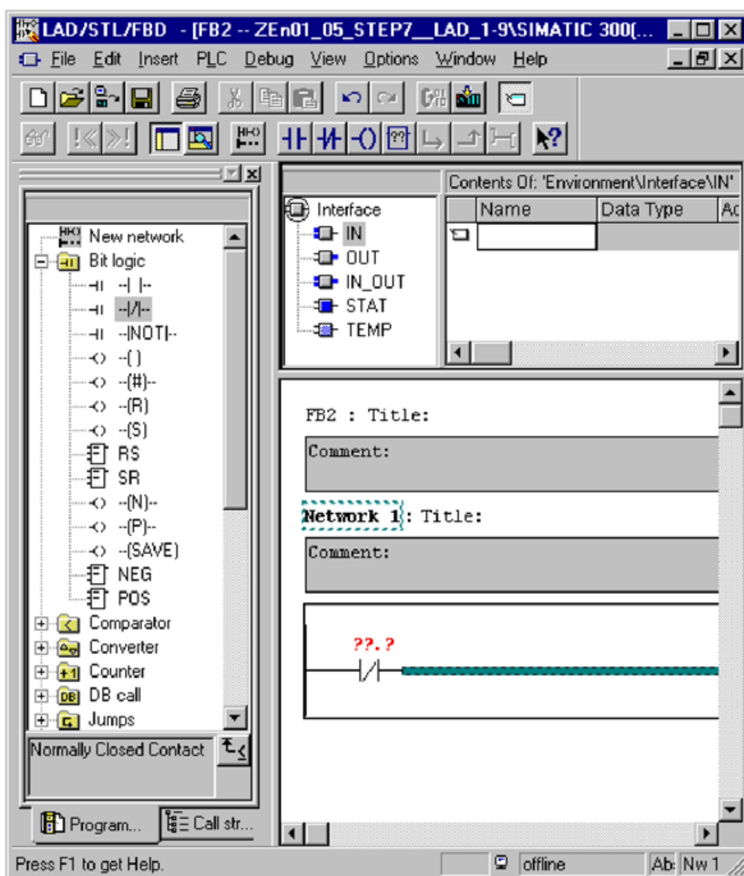
读/写访问权分配如下：

- 离线编辑：
  当用户试图打开一个块/源文件时，会检查该用户是否对该块有“写”访问权。如果块/源文件已经打开，用户只能进行复制。如果用户希望存盘，系统会要求是否想要覆盖原块，还是存到一个新的名下。
- 在线编辑：
  当用户通过组态连接打开一个在线块时，对应的离线块禁止使用，以保护离线块不会被同时修改。

10.1.5 程序元素表中的指令集

程序元素表中提供一系列LAD和FBD中的编程元素，以及已经声明的多重背景、已经编写的逻辑块和程序库中的各逻辑块。目录由菜单命令View > Tables来访问。用菜单命令Insert > Program Elements，将程序元素插入到程序指令部分中。

LAD中程序元素表举例

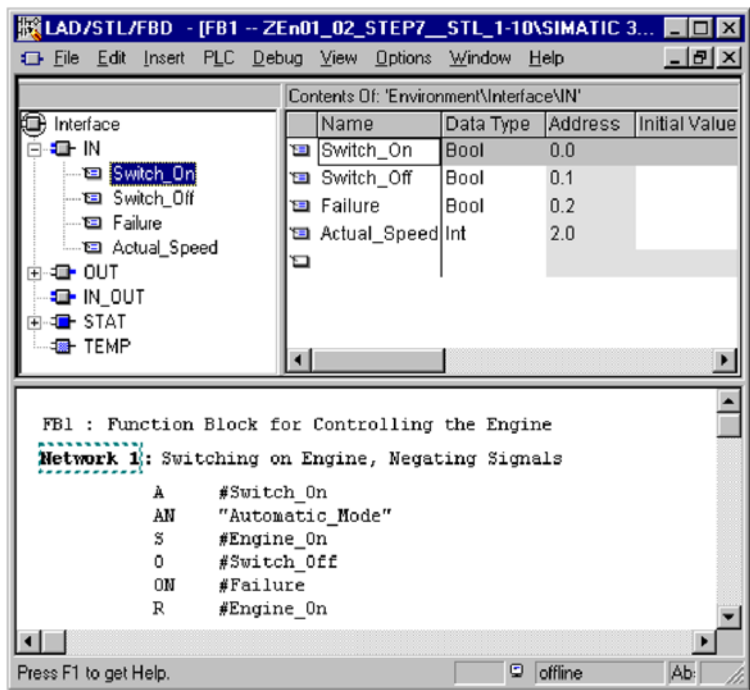


10.2 编辑变量声明表

10.2.1 在逻辑块中的变量声明

当用户打开一个逻辑块时，在窗口的上半部分为变量声明表，下半部分为程序指令部分，用户在下半部分编写逻辑程序。

例如：STL中变量声明表和程序指令部分



在变量声明表中，用户声明在本块中专用的变量包括块的形参和参数的系统属性。这具有以下作用：

- 声明变量后，在本地数据堆栈中为临时变量保留一个有效存贮空间，对于功能块，还要为背景数据块的静态变量保留空间。
- 当设置输入、输出和输入/输出类型参数时，用户还要在程序中声明块调用的“接口”。
- 当用户给某功能块声明变量时，这些变量（临时变量除外）也在与功能块关联的背景数据块中的数据结构中声明。
- 通过设置系统特性，用户为信息和连接组态操作员接口功能分配特殊的属性，以及参数的过程控制组态。

10.2.2 变量声明表与指令部分之间的关系

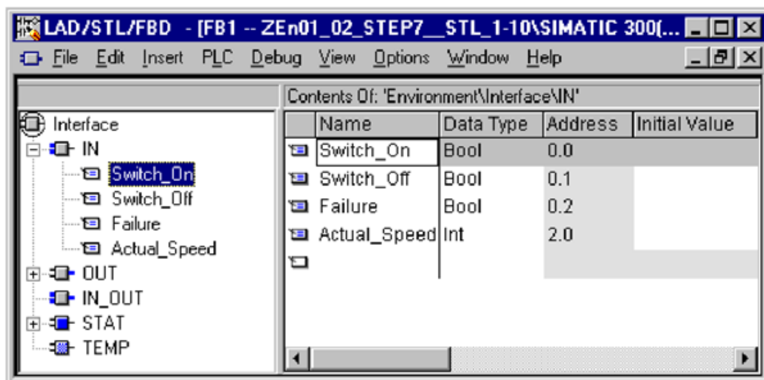
逻辑块中的变量声明表和指令部分是紧密联系的，因为在指令部分的程序中要用到变量声明表中的名称。因此，在变量表中的任何变化都将影响整个指令部分的程序。

变量声明表中的改动	指令部分的反应
重新正确输入	使用以前没在变量表中声明的变量，造成无效语句，现在变为有效
变量名改变但类型没变	所有地方的符号立即更改为相应的新名称
正确的变量名改为无效的变量名	指令保持不变
无效的变量名改为正确的变量名	如果有无效语句，可变为有效
类型改变	如有无效语句，可变为有效 如有有效语句，可变为无效
删除一个程序中使用的变量（符号名）	有效语句变为无效

文字注释内容的修改、一个新变量的不正确输入、改变初始值或删除一个没用的变量，对指令部分没有影响。

10.2.3 变量声明表的结构

变量声明表窗口包括变量概述和变量详细说明。



当打开一个新的程序块后，将显示一个缺省的变量表。它只列出了与块有关联的声明类型(输入、输出、输入/输出、静态、动态变量)，可以编辑这个缺省的变量声明表。

在附录表“逻辑块的本地数据分配数据类型”中可以发现所有本块数据区中允许的数据类型。

10.3 变量声明表中的多重背景

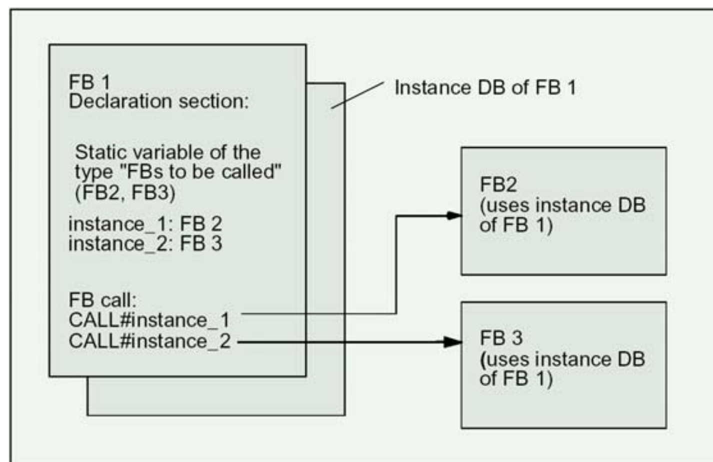
10.3.1 使用多重背景

如果用户希望或不得不用有限的几个数据块存放背景数据以提高S7 CPU中的性能（例如：存储能力）是可能的。如果在功能块中调用其它的功能块（FB），不需要它们自己的（额外的）背景数据块。

解决的方法如下：

- 在调用功能块的变量声明表中，将被调用的功能块做为静态变量参数。
- 在该功能块中，调用其它功能块不带有（额外的）背景数据块。
- 这就将背景数据都压缩在一个背景数据块中，用户能够更有效地利用数据块的资源。

下面的例子所示为：FB2和FB3使用调用它们的FB1的背景数据块。



唯一的要求：用户必须“告诉”调用功能块，哪个背景需要调用以及这些背景的类型（FB是什么?）。这些细节必须在调用功能块的参数声明窗口输入。被调用的功能块在数据区中至少要有一个变量或参数（VAR_TEMP不能用）。

10.3.2 声明多重背景的规则

多重背景的声明有下列规则：

- 只有在版本2以上的STEP 7中生成的功能块（参看功能块的属性中的块特性），才可能声明多重背景。
- 为了声明多重背景，功能块必须设置为有多重背景能力（在STEP 7中缺省设置），可在编辑器中用**Options > Customize**取消。

- 必须有一个背景数据块分配给声明了多重背景的功能块。
- 多重背景只能声明为静态变量（声明类型为“Stat”）。

注意

- 用户也可以为系统功能块生成多重背景。
- 如果功能块生成时不具备多重背景的能力，而又需要增加这个功能，可以用该功能块产生一个源文件，然后将块属性中的CODE_VERSION1删除，再重新编译该功能块即可。

10.3.3 在变量声明窗口中输入多重背景

1. 打开功能块，在该功能块中将调用下一级功能块。
2. 如果不想给被用调的功能块使用背景数据块，可以为这些功能块在调用它们的功能块变量声明表中定义一个静态变量。
   - 在变量表中选择“STAT”。
   - 在变量视窗的“Name（名）”栏中为FB的调用输入一个名字。
   - 在“Data Type（数据类型）”栏中输入需要调用的功能块作为一个绝对地址或用它的符号名。
   - 可以在注释栏输入任何需要的注释。

在程序部分调用

如果声明了多重背景，可以调用FB和无需指定一个背景DB。

例如：如果定义了静态变量 "Name: Motor_1 , Data type: FB20"，则可按下述方法调用背景：

Call Motor_1 // 调用FB20，无背景数据块

10.4 编辑语句和文字注释时的注意事项

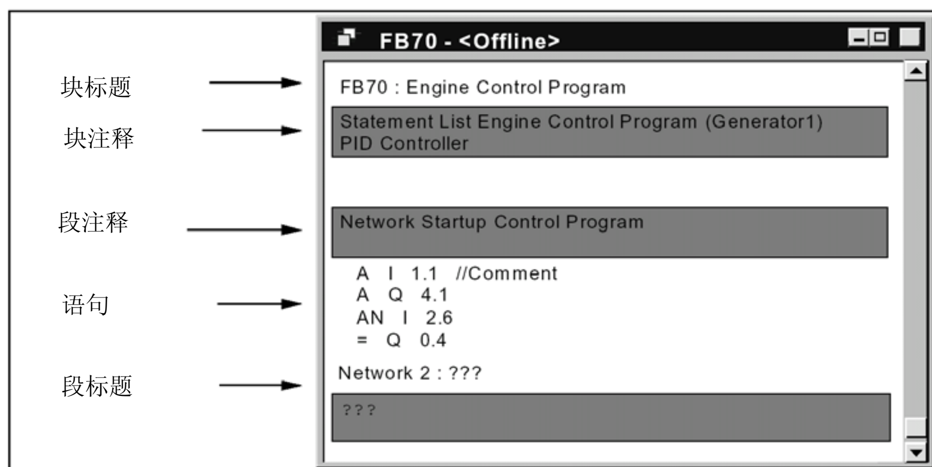
10.4.1 程序指令部分的结构

在程序指令部分，选择编程语言，输入相应的语句按顺序编写逻辑块中的程序。当语句输入进去时，编辑器立即起动句法检查，发现的错误用红色和斜体显示。

逻辑块的程序指令部分通常由若干网络段组成，而这些网络段又由一系列语句组成。

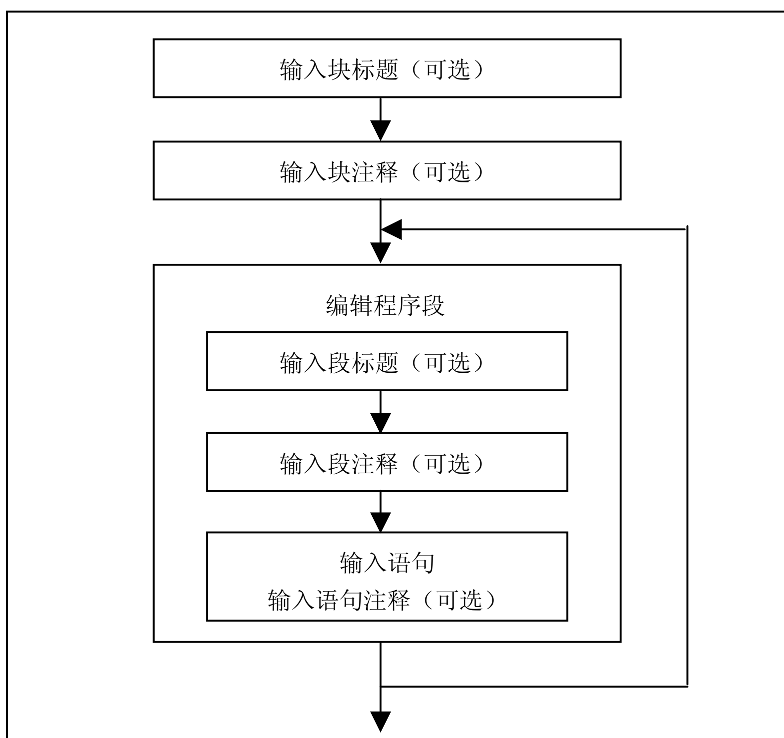
在程序指令部分，用户可以编辑块标题、块注释、段标题、段注释和各程序程序段中的语句行。

用STL编程语言的程序指令部分结构



10.4.2 输入语句的步骤

用户可以按任意的顺序来编辑程序。当第一次编程时，建议你按如下步骤进行。



可以插入新的程序段或覆盖原有的程序段进行修改，用执行键（INSERT）切换两种修改状态。

10.4.3 在程序中输入共享符号

使用菜单命令Insert > Symbol，用户可在程序的指令部分插入符号，当光标位于字符串的开始、结束或中间时，如果符号已经存在，则总是以小短线“-”开始。如果用户修改字符串，则该选择也在符号表中更新。

一个字符串开始和结尾的分开标志是，例如，空格、句号、逗号。在共享符号中没有分开标志。

按如下步骤输入符号：

1. 在程序中输入所需符号的第一个字母。
2. 同时按下CTRL和 J 键，显示符号表。第一个符号按输入的字母开始。
3. 按RETURN键输入符号或选择另一个符号。

然后，使用引号圈起来的符号代替第一个字母。

通常下列情况实现：如果光标位于一个字符串的开始、结尾或中间，当插入一个符号时，该字符串被带引号的符号所替代。

10.4.4 块和段的标题与注释

注释可使程序易于阅读，因此，也使得程序调试和故障诊断容易进行且更有效率。它们是程序文档中的重要组成部分，应该加以利用。

在梯形图、功能块图和语句表程序中的文字注释

可用下列文字注释：

- 块标题：块的标题（最多64个字符）。
- 块注释：整个逻辑块的文字说明，例如：该块的目的。
- 段标题：段的标题（最多64个字符）。
- 段注释：单个段的功能说明。
- 变量详细窗口中的注释栏：所声明的本块数据的说明。
- 符号注释：地址注释，在符号表中已经定义了符号名。用户可以用菜单命令View > Display with > Symbol Information，显示这些注释。

在逻辑块的程序部分，用户可以输入块标题、段标题、块注释或段注释。

块标题或段标题

要想输入块标题或段标题，需把光标放在“Title”的位置，向右输入块名称或段名（例如Network1：Title）。当输入标题时打开一个文字输入框，最长可输入64个字符。

1. Network 2: Title ← 鼠标点击

2. Network 2: 在此显示程序段标题

块注释是整个逻辑块的注释，在块注释中，可以注释块的功能。段注释是具体段的注释，可以详细说明段。

自动显示程序段标题，选择**Options > Settings**并在“General”标签中点击选项“Automatic Assignment of Network Title”。

块注释和段注释

1. ? ? ? ← 鼠标点击

2. Comment for network or block

使用菜单命令**View > Display with > Comments**，显示或不显示灰色的注释域。双击注释域打开文字输入框，可输入注释要点。当输入标题时打开一个文字输入框，最长可输入64个字符。

10.4.5 输入块注释和网络段注释

1. 使用菜单命令**View > Display with > Comments**激活注释。在菜单命令前可以看到检查标记。
2. 点击鼠标，将光标指向块或网络段下面的灰色栏内。此时灰色注释栏变为白色，并出现一个文本框。
3. 在打开的文本框中输入注释。每个块注释和网络段注释只允许64K字节。
4. 在文本框外点击鼠标、按TAB键或使用SHIFT+TAB组合键可退出文本框。
5. 若再次选择菜单命令**View > Display with > Comments**，则可以关掉注释显示。

10.4.6 使用网络段模板

当编写程序块时，如果多次使用网络段，可以将这些网络段存储在库中作为网络段模板。在建立网络段模板前，该库必须存在。

建立网络段模板

如果需要，在SIMATIC Manager其中建立一个新库。选择**Insert > Program > S7 Program**将一个程序插入库中。

1. 打开包含网络段的程序块，从这里可以建立一个网络段模板。
2. 在打开的块中，按照需要用通配符替换标题、注释或地址。可以使用字符串%00至%99作为通配符。地址通配符以红色显示，不会对原有的程序块产生影响，因为在创建网络段模板后，不会存储该程序块。
3. 选择需要包含在网络段模板的“Network <No.>”。
4. 选择**Edit > Create Network Template**。
5. 在显示的对话框中为每个通配符输入一个注释。
6. 点击“OK”
7. 在网络段模板库中选择S7程序的Source file folder(源文件夹)并为网络段模板输入名称。
8. 点击“OK”确认输入。网络段模板存储在所选择的库中。
9. 可以不存盘退出该程序块。

在一个程序中插入一个网络段模板

1. 打开一个程序段，插入一个新的程序段。
2. 在打开的程序段中，点击网络段，在此之后按照网络段模板插入一个新的网络段。
3. 使用**Insert > Program Elements**打开程序单元。
4. 在目录中的相应库中打开“S7 Program”文件夹。
5. 双击网络段模板。
6. 插入对话框，在网络段模板中输入所需替换的通配符。
7. 点击“OK”，在当前的网络段后面将插入一个网络段模板。

10.4.7 程序指令中错误搜索功能

在程序指令部分的错误，由于是红颜色，很容易识别。编辑器提供两种搜索功能**Edit > GoTo > Previous Error/Next Error**，用于查找屏幕可见部分以外的错误。

搜索错误的范围超过一个程序段，这就意味着在整个程序指令部分搜索，不只是一个段，或当前屏幕能看见的范围。

如果用菜单命令**View >Status Bar**激活状态条，找到的错误显示的地方。

用户也可在修状态下修改错误和进行改进。用插入键（INSERT）可切换插入状态和修改状态。

10.5 在程序代码区编辑梯形图（LAD 编程）

10.5.1 梯形逻辑编程的一些设置

设置梯形逻辑布局

用户可以设置在梯形逻辑编程语言中生成程序的布局。所选的格式（A4窄幅/宽幅/最大尺寸）将影响一行中所能显示的梯形元素的数目。

1. 选择菜单命令Options > Customize。
2. 在出现的对话框中选择“LAD/FBD”标签。
3. 在“Layout（布局）”列表框中，选择所需的格式。输入所需格式的大小。

设置打印

如果希望打印梯形程序指令部分，用户应在开始编程之前，设置合适的打印纸格式。

在“LAD/FBD”标签中的设置

用菜单命令Options > Customize，可访问“LAD/FBD”标签，在该标签中可以进行基本设置，例如布局和地址域宽。

10.5.2 输入梯形逻辑元素的规则

用户在《S7-300/400梯形逻辑编程参考手册》一书中，或梯形逻辑在线帮助中，可以找到有关梯形逻辑编程语言表示法的描述。

一个梯形程序段中，可以有多个分支，每条分支上可有多个元素。所有的元素和分支都必须连接；左边的能源轨不作为连接（IEC 1131-3）。

当用户编写梯形程序时，必须遵守编程规则。如果有错误发生，会有信息提示。

结束梯形程序段

每个梯形程序段都必须以输出线圈或功能框结束，下列的梯形元素不能用于程序段结束：

- 比较框
- 中间输出结果的线圈 _/(#)_/
- 上升沿 _/(P)_/ 或下降沿 _/(N)_/ 线圈

功能框的位置

用于功能框连接的分枝起始点必须总是左边的能源轨，在该功能框前的分支上可以有逻辑操作或其它功能框。

线圈的位置

线圈自动位于程序的最右端，在这个位置上形成分支的终点。

下列情况除外：用于中间结果输出的线圈_/(#)_/ ，以及上升沿线圈 _/(P)_/ 或下降沿线圈 _/(N)_/ ，都不能置于分支的最左端或最右端，也不允许放在平行分支上。

有些线圈需要布尔逻辑操作，而有些线圈一定不能进行布尔逻辑操作。

- 允许布尔逻辑的线圈为：
  - 输出 _/()，置位输入_/(S)，复位输入 _/(R)
  - 中间结果输出_/(#)_/，上升沿 _/(P)_/，下降沿 _/(N)_/
  - 所有的计数器和定时器线圈
  - 逻辑非跳转 _/(JMPN)
  - 主控继电器接通 _/(MCR<)
  - 将RLO存入BR存贮器_/(SAVE)
  - 返回_/(RET)
- 不允许布尔逻辑的线圈：
  - 主控继电器激活 _/(MCRA)
  - 主控继电器取消_/(MCRD)
  - 打开数据块_/(OPN)
  - 主控继电器关_/(MCR>)

其它的线圈既可以用布尔逻辑操作也可以不用。

下列线圈一定**不能用于平行输出**：

- 逻辑非跳转 _/（JMPN）
- 跳转 _/（JMP）
- 从线圈调用 _/（CALL）
- 返回_/（RET）

使能输入/使能输出

功能框的使能输入端“EN”和使能输出端“ENO”可以连接使用，也可以不用。

删除和覆盖

如果分支中只有一个元素，当删除这个元素时，整个分支也同时删掉。

当一个功能框删除时，除主分支外，该功能框的所有布尔输入分支都将删除。

覆盖模式可用于简单的同类型元素的覆盖。

平行分支

- 从左到右画一个或（OR）分支。
- 向下打开平行分支，向上闭合。
- 选择某一梯形元素后，总是可以打开一个平行分支。
- 在选择的梯形元素之后，总是可以闭合一个平行分支。

- 删除一个平行分支，将删掉该分支上的所有元素。当分支上最后一个元素删除，则分支自动删除。

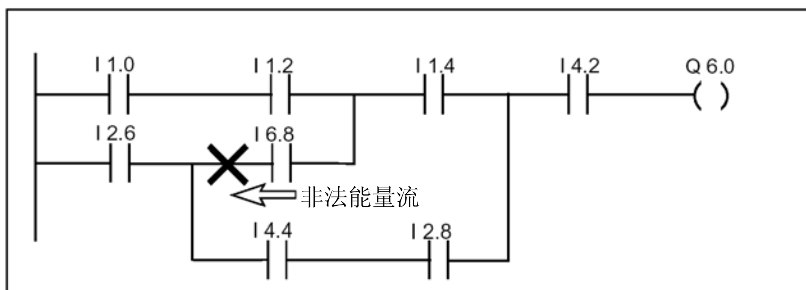
常数

二进制连接不能赋予常数(例如：TRUE或FALSE)。取而代之，用BOOL数据类型地址。

10.5.3 梯形图中的非法逻辑操作

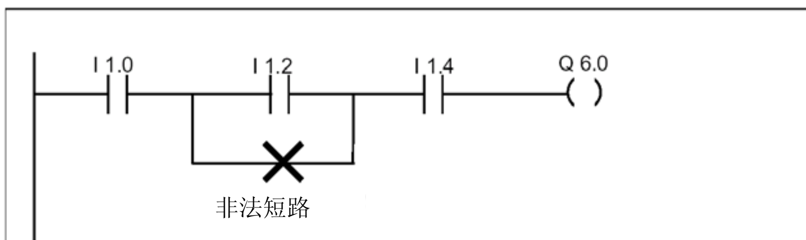
能量流从右到左

不允许生成使能量流向相反方向的分支。下图所示为：当I 1.4的信号状态为“0”时，能量流经I 6.8的方向是从右到左，这是不允许的。



短路

不允许生成造成短路的分支。下图所示为举例：



10.6 在程序代码区编辑功能块图（FBD）

10.6.1 功能块图编程的一些设置

设置功能块图布局

用户可以设置在功能块图编程语言中生成程序的布局。所选的格式（A4窄幅/宽幅/最大尺寸）将影响一行中所能显示的FBD元素的数目。

1. 选择菜单命令Options > Customize。
2. 在出现的对话框中选择“LAD/FBD”标签。
3. 在“Layout（布局）”列表框中，选择所需的格式。输入所需格式的大小。

设置打印

如果希望打印FBD的程序指令部分，用户应在开始编程之前，设置合适的打印纸格式。

在“LAD/FBD”标签中的设置

用菜单命令Options > Customize可访问“LAD/FBD”标签，在该标签中可以进行基本设置，例如布局和地址域宽。

10.6.2 输入FBD元素的规则

用户在《S7-300/400功能块图编程参考手册》一书中，或FBD在线帮助中，可以找到有关FBD编程语言表示法的描述。

一个FBD程序段中，可以有多个元素，所有的元素都必须连接（IEC1131-3）。

当用户编写FBD程序时，必须遵守编程规则，如果有错误发生，会有信息提示。

输入和编辑地址与参数

当插入一个FBD元素时，符号“???”和“…”用作地址和参数的标记符号。

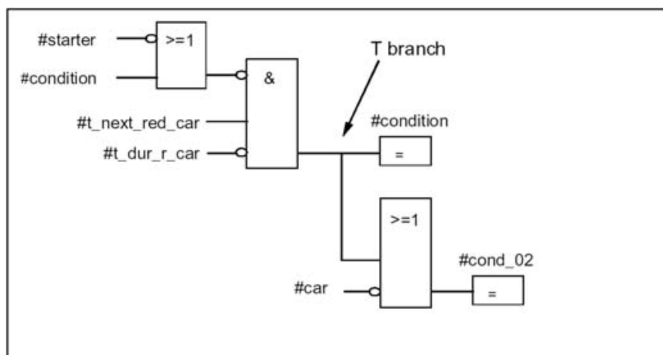
- 红色的标记符号“???”表示其地址和参数必须修改。
- 黑色的符号“…”表示其地址和参数可以修改。

如果用户将鼠标点在标记符号上，可显示相应的数据类型。

功能框的位置

标准功能框（触发器、计数器、定时器、算术操作等）可插在二进制逻辑操作框（&（与），>=1（或），XOR（异或））的中间。比较功能框除外。

在一个程序段内不允许不同逻辑操作的不同输出，然而，用户可以利用T型分支将一串逻辑操作分别赋值给不同的输出。下图所示为一个程序段中有两个赋值输出。



下列功能框只能放在逻辑串的最右边，在那结束该逻辑串。

- 设置计数器值
- 加计数器参数赋实参（实际参数），减计数器形参赋实参
- 脉冲定时器参数赋值及起动，扩展脉冲定时器参数赋值及起动
- 接通延迟/断开延迟定时器的参数赋值和起动

有些功能框要求布尔逻辑操作，而有些功能框一定不能用布尔逻辑操作。

允许布尔逻辑操作的功能框：

- 输出，置位输出，复位输出 _/[R]
- 中间结果输出_/[#]_/，上升沿_/[P]_/，下降沿_/[N]_/
- 所有的计数器和定时器功能框
- 逻辑非跳转 _/[JMPN]
- 主控继电器接通 _/[MCR<]
- 将RLO存入BR存贮器 _/[SAVE]
- 返回 _/[RET]

不允许布尔逻辑的功能框：

- 主控继电器激活[MCRA]
- 主控继电器取消[MCRD]
- 打开数据块[OPN]
- 主控继电器断开[MCR>]

所有其它功能框既可用布尔罗辑，也可不用。

使能输入/使能输出

功能框的使能输入端“EN”和使能输出端“ENO”可以连接使用，也可以不用。

删除和覆盖

当一个功能框删除时，除主分支外，该功能框的所有布尔输入分支都将删除。

覆盖模式可用于简单的同类型元素的覆盖。

常数

二进制连接不能赋予常数(例如：TRUE或FALSE)。取而代之，使用BOOL数据类型地址赋值。

10.7 在程序指令部分编辑 STL 语句

10.7.1 语句表编程的设置

记忆性设置

用户可以选择两种记忆性设置：

- German（德语）
- English（英语）

用户在打开逻辑块之前，在SIMATIC Manager中通过菜单命令**Options > Customize**中的“Language（语言）”标签设置记忆性。在编辑块时，不能改变记忆码。

在对话框中，编辑块属性。

在编辑器中，用户可以打开许多块，需要时交替地进行编辑。

10.7.2 输入STL语句的规则

用户在《S7-300/400语句表编程参考手册》一书中，或STL在线帮助中，可以找到有关语句表编程语言的描述。

当输入STL语句时，必须注意以下基本规则：

- 逻辑块编程的顺序很重要。在调用某逻辑块之前，该块必须存在。
- 每条语句都是由跳转标签（可选）、指令、地址和文字注释（可选）组成。
  例如：　M001：A　　　I 1.0　　　//Comment
- 每条语句占一行。
- 逻辑块中最多有999个程序段。
- 每个程序段最多将近2000行左右。如果用户使用放大或缩小功能，相应的行数多些或少些。
- 当输入指令或绝对地址时，不区分大小写。

10.8 更新块调用

可以在“LAD/STL/FBD—Programming S7 Blocks”编辑器中，使用菜单命令Edit > Block Call > Update，自动更新那些由于下列接口发生变化而变为非法的块调用或用户定义数据类型：

- 插入新形参
- 删除形参
- 修改形参名
- 修改行参类型
- 修改形参的顺序

当用户进行形式参数和实际参数赋值时，必须按顺序遵循下列规则：

1. 参数名相同：

   如果形参名保持不变，实参自动赋值。

   特例：在梯形图和功能块图中，预先连接的二进制输入参数只有在数据类型（BOOL）相同时才自动赋值。如果数据类型改变，则预先连接变为开路分支。

2. 参数的数据类型相同：

   当同名参数已经赋值完成之后，还未赋值的实参将赋值给那些与“旧”形参数据类型相同的形参。

3. 参数位置相同：

   当用户按规则1和2执行完之后，还没有赋值的实参，现在按照在“旧”接口的参数位置，赋给形参。

4. 如果按照上面三条规则，还有实参不能赋值，它们将被删除，或在梯形图和功能图中预先连接的二进制信号，将变为开路分支。

当执行完这个功能之后，请检查在变量声明表和程序指令部分做的修改。

10.8.1 改变接口

也可以用incremental Editor修改用STEP 7 V5编写的离线块接口：

1. 确信所有的块均已用STEP 7 V5编译过。
2. 修改相关块的接口。
3. 一个一个地打开所有调用的块 – 相应的调用以红色显示。
4. 选择菜单命令 Edit > Block Call > Update。
5. 再次生成相关的背景数据块。

注意：

- 改变在线打开的程序块接口时可能会导致CPU进入STOP模式。
- 重新进行块调用。首先修改所调用块的块号，然后执行Rewire功能以匹配所进行的调用。

10.9 逻辑块存盘

输入新生成的程序块或在编程器中修改程序指令部分或变量声明表之后，用户必须将相应的块存盘，使该数据存储到编程器的硬盘中。

将逻辑块存入到编程器的硬盘中：

1. 打开需要存盘的块。
2. 选择下列菜单命令之一：
   - File > Save，将块存在同一名下。
   - File > Save As，将块存在不同的S7用户程序下或不同的名下。在出现的对话框中输入新的路径或新的块名。

在以上两种情况中，只有当逻辑块中没有句法错误时才能存盘。当生成逻辑块出现句法错误时，会立即识别并用红颜色显示出来。这些错误必须在存盘之前修改。

注意

- 用户也可以在SIMATIC Manager中（比如，用拖放功能）将逻辑块或源文件存到其它项目或程序库中。
- 用户在SIMATIC Manager中，只能将块或整个用户程序存到存储卡中。
- 如果某些较大的逻辑块在存盘或编译时出现问题，用户应重新识别项目。在SIMATIC Manager中，使用菜单命令File > Reorganize 进行识别。然后，再尝试一次存盘或编译。

11 数据块的生成

11.1 有关生成数据块的基本信息

数据块（DB）可用于存贮设备或生产线中可访问的值。与使用编程语言梯形图、语句表或功能块图之一逻辑块不同，数据块只有变量声明部分，这就意味着，在这里没有程序指令部分，所以也就没有程序代码。

当用户打开数据块时，既可用声明表形式显示，也可用数据显示形式浏览数据块，用户可以使用菜单命令View > Declaration View 和 View > Data View切换两种显示状态。

声明表显示状态

使用声明表显示状态可以：

- 浏览或确定共享数据块的数据结构。
- 浏览与用户定义数据类型（UDT）关联数据块的数据结构。
- 浏览功能块（FB）背景数据块的数据结构。

与功能块或用户定义数据类型相关的数据块的结构不能修改。如需修改，必须修改相应的FB或UDT，然后再生成一个新数据块。

数据显示状态

如果用户想修改数据，就应用数据显示状态。在数据显示状态中，用户可以选择显示，输入或改变每个元素的实际值。在数据块的数据浏览中，复合数据类型变量的各元素，分别用其全名列出。

背景数据块与共享数据块之间的区别

共享数据块不附属于任何逻辑块。它含有生产线或设备所需的值，并可以在程序的任何点直接使用。

背景数据块直接附属于某逻辑块，例如：功能块。背景数据块中所含数据为功能块的变量声明表中所存数据。

11.2 数据块中声明表形式显示

对于不能共享的数据块，在声明表显示状态不能进行修改。

列	解　　释
地址（Address）	当用户结束变量声明的输入时，STEP 7自动分配并显示地址
声明类型（Declaration）	本栏只在背景数据块中显示。表明在功能块的变量声明表中各变量是如何声明的： • 输入参数（IN） • 输出参数（OUT） • 输入/输出参数（IN_OUT） • 静态数据（STAT）
名称　（Name）	这里输入给每个变量的符号名
数据类型　（Type）	输入用户赋给变量的数据类型（BOOL，INT，WORD，ARRAY，等）。变量可以有基本数据类型，复合数据类型，或用户声明数据类型
初值　（Initial Value）	这里可输入初值，如果用户没有输入，则软件根据所输入的数据类型给出缺省值。 如果用户没有给变量声明实际值，当数据块第一次存盘时，初值将用做实际值。 请注意：初值不能下载到CPU中。
注释　（Comment）	输入对变量文档有帮助的注释，最多可以有80个字符

11.3 数据块中的数据总览

数据显示状态为用户显示该数据块中所有变量的实际值。用户只能在数据显示状态形式中改变这些值。对所有的共享数据块来说，数据显示状态的表格表示都是相同的。对于背景数据块，还增加一个“参数声明（Declaration）”栏。

对于用复合数据类型或用户定义数据类型的变量，在数据浏览时，所有的元素都占有自己的一行，并有其完整的符号名。如果元素位于背景数据块的IN_OUT区，则指针指向“实际值（Actual value）”栏中的复合数据类型或用户定义数据类型。

数据显示状态中有下列栏目显示：

栏	解　　释
地址（Address）	STEP 7自动为变量分配并显示地址
声明类型（Declaration）	本栏只在背景数据块中显示，表明在功能块的变量声明表中各变量是如何声明的： • 输入参数（IN） • 输出参数（OUT） • 输入/输出参数（IN_OUT） • 静态数据（STAT）

栏	解　　释
名称（Name）	在变量声明时给的符号名。在数据浏览中用户不能编辑该域。
数据类型（Type）	显示变量所声明的数据类型。 对于共享数据块，这里只列出基本数据类型，因为在数据浏览中，复合数据类型变量或用户声明数据类型要分别列出元素。 对于背景数据块，也显示参数类型，对于用复合或用户声明数据类型的in/out参数（IN_OUT），指针指向“实际值”栏中的数据类型。
初值（Initial Value）	用户为变量输入的初值，如果用户没有输入，则软件会根据所声明的数据类型给出缺省值。 如果用户没有给变量声明实际值，当数据块第一次存盘时，初值将用做实际值。 请注意：初值不能下载到CPU中。
实际数值 （Actual Value）	离线：打开数据块时最后保存的变量数值（即使是在线打开的数据块，也不刷新该显示）。 在线：将显示打开数据块时的实际数值，但不会自动刷新。按动F5，可以刷新该窗口。 如果不属于复合或用户定义数据类型的输入/输出参数（IN_OUT），可以编辑该区域。所有的值必须与数据类型匹配。
注释（Comment）	对变量进行说明所输入的注释。在数据浏览中用户不能编辑该域。

11.4　数据块的编辑与存盘

11.4.1　输入共享数据块的数据结构

如果用户打开一个不是使用功能块或用户定义数据类型生成的数据块，则可以在该数据块的声明表显示方式中声明其结构。对于不能共享的数据块，在声明表显示方式中不能进行修改。

1. 打开一个共享数据块，该数据块与UDT或FB无关。
2. 如果显示方式没有设置，选择声明表显示方式进行显示。
3. 按下列信息，通过填写表格来声明数据块的结构。

对于不能共享的数据块，声明表显示状态不能修改。

列	解　　释
地址（Address）	当用户结束变量声明的输入时，STEP 7自动分配并显示地址
名称（Name）	给每个变量的命名符号名
数据类型 （Type）	赋给变量的数据类型（BOOL，INT，WORD，ARRAY，等）。变量可以有基本数据类型，复合数据类型，或用户声明数据类型
初值 （Initial Value）	这里可输入初值，如果用户没有输入，则软件根据所输入的数据类型给出缺省值。如果用户没有给变量声明实际值，当数据块第一次存盘时，初值将用做实际值

列	解　　释
注释 （Comment）	输入对变量文档有帮助的注释。最多可以有80个字符。

11.4.2　输入并显示与FB有关的数据块（背景DB的数据结构）

输入

当用户将数据块与某一功能块相连时（背景DB），则功能块的变量声明表决定该数据块的结构。任何修改只能在相关的功能块中进行。

1. 打开相关的功能块（FB）。
2. 编辑该功能块的变量声明表。
3. 再生成背景数据块。

显示

在背景数据块的声明表显示方式中，用户可以显示功能块的变量是如何声明的。

1. 打开数据块。
2. 如果显示方式没有设置，选择声明表显示方式进行显示。
3. 参看下表中有关声明表的更多信息。

对于不能共享的数据块，在声明表显示方式下不能进行修改。

列	解　　释
地址（Address）	STEP 7自动为变量分配并显示地址。
参数类型 （Declaration）	本栏表明在功能块的变量声明表中各变量是如何声明的： • 输入参数（IN） • 输出参数（OUT） • 输入/输出参数（IN_OUT） • 静态数据（STAT） 所声明的功能块临时局域数据不位于背景数据块中。
名称（Name）	在功能块变量声明表中给出的符号名。
数据类型 （Type）	显示功能块的变量声明表中给出的数据类型。变量可以有基本数据类型，复合数据类型，或用户声明数据类型。 如果在功能块中调用其它功能块，必须在静态变量中声明，也可以作为复合数据类型声明一个功能块或一个系统功能块（SFB）。
初值 （Initial Value）	用户在功能块的变量声明表中输入初值，如果没有输入，则软件给出缺省值。 当数据块第一次存盘时若用户没有明确地声明实际值，则初值将被用于实际值。
注释（Comment）	在功能块的变量声明表中输入的注释，以便对数据元素文字说明。用户不能编辑该域。

注意

对于分配给某功能块的数据块，用户只能编辑变量的实际值。用户必须在数据块的声明表显示方式中输入变量的实际值。

11.4.3 输入用户定义的数据类型（UDT）的数据结构

1. 打开用户定义数据类型（UDT）。
2. 如果该显示没有设置，以声明表方式显示。
3. 通过声明变量的顺序、变量的数据类型以及初值确定UDT的结构。下表中给出所需的信息。
4. 用TAB或RETURN键退出某行，来完成该行的变量输入。

列	解　　释
地址（Address）	当用户结束变量声明的输入时，STEP 7自动分配并显示地址。
名称（Name）	这里输入给每个变量的符号名。
数据类型（Type）	输入用户赋给变量的数据类型（BOOL，INT，WORD，ARRAY，等）。变量可以有基本数据类型，复合数据类型，或用户声明数据类型。
初值（Initial Value）	这里可输入初值，如果用户没有输入，则软件根据所输入的数据类型给出缺省值。所有的值必须与数据类型匹配。如果用户没有给变量声明实际值，当第一次用户定义数据类型或一个变量，或一个数据块的背景存盘时，初值将用做实际值。
注释（Comment）	输入对变量进行文字说明的注释，最多可以有80个字符。

11.4.4 输入并显示与UDT相关的数据块结构

输入

当用户用UDT生成数据块时，UDT的数据结构就决定了该数据块的结构。任何修改都只能在相关的用户定义数据类型（UDT）中进行。

1. 打开用户定义数据类型（UDT）。
2. 编辑用户定义数据类型的结构。
3. 再生成数据块。

显示

用户只能在数据块的声明表显示方式中，显示用户定义数据类型中的变量。

1. 打开数据块。
2. 如果该显示没有设置，以声明表方式显示。
3. 参看下表中有关声明表的更多信息。

在声明表显示方式下不能进行修改。任何修改都只能在相关的用户定义数据类型（UDT）中进行。

列	解　　释
地址（Address）	STEP 7自动为变量分配并显示地址。
名称（Name）	在用户数据类型的变量声明表中给出的符号名。
数据类型（Type）	显示在用户声明数据类型的变量声明表中给出的数据类型。变量可以有基本数据类型，复合数据类型，或用户声明数据类型。
初值（Initial Value）	用户在用户声明数据类型中为变量输入的初值，如果用户不想输入，则软件给出缺省值。 当数据块第一次存盘时若用户没有明确地声明实际值，则初值将被用于实际值。
注释（Comment）	在用户声明数据类型的变量声明表中输入的注释，用于对数据元素进行文字说明。

注意

对于用UDT生成的数据块，用户只能编辑变量的实际值。用户必须在数据块的声明表显示方式中输入变量的实际值。

11.4.5　在数据显示方式下编辑数据值

只有在数据块的数据显示方式下，才可能编辑实际值。

1. 如果需要，使用菜单命令View > Data View，将显示切换到数据显示方式。
2. 在“实际值（Actual Value）”一栏中输入各数据元素的实际值。实际值必须与这些数据元素的数据类型相匹配。

任何不正确的输入（例如，如果输入的实际值与数据类型不匹配，在编辑时能立即识别，并显示为红颜色）。这些错误必须在数据块存盘之前改正过来。

注意

一旦数据块存盘，数据值的任何变化都将保存。

11.4.6　将数据值恢复为初始值

只有在数据块的数据显示方式下，才可能将数据值恢复为初始值。

1. 如果需要，使用菜单命令View > Data View，将显示切换到数据显示方式。
2. 选择菜单命令Edit>Initialize Data Block来进行。

所有的变量又恢复了初始值，这意味着所有变量的实际值被初始值所替代。

注意

一旦数据块存盘，数据值的任何变化都将保存。

11.4.7 存储数据块

在编程器数据库中，新生成的块或在数据块中修改了数据值，用户都需要存盘，这样该数据可以写到编程器的硬盘中。

将逻辑块存入到编程器的硬盘中：

1. 打开块的编辑窗口。
2. 选择下列菜单命令之一：
   - **File > Save**，将块存在同一名下。
   - **File>Save as**，将块存在不同的 S7 用户程序中或不同的项目名下。在出现的对话框中输入新的路径或新的块名。对于数据块，用户不能使用DB0，因为这个号码被系统占用。

在以上两种情况中，只有当逻辑块中没有语法错误时才能存盘。当生成逻辑块时，出现语法错误，会立即识别并用红颜色显示出来。这些错误必须在存盘之前修改。

注意

- 用户也可以在SIMATIC Manager中（比如，用拖放功能）将逻辑块或源文件存到其它项目或程序库中。
- 用户在SIMATIC Manager中，只能将块或整个用户程序存到存储卡中。
- 如果某些较大的逻辑块在存盘或编译时出现问题，用户应重新识别项目。在SIMATICManager 中，使用菜单命令**File > Reorganize** 进行识别，然后，再试一次存盘或编译。

12 数据块的参数赋值

对数据块的参数赋值功能可以不在LAD/STL/FBD程序编辑器中进行：

- 不用调用整个数据块，可以修改背景数据块的实际值并下载到PLC中。
- 在线监视背景数据块。
- 用“S7_Techparam”系统属性(技术功能)可以方便地对背景数据块和多重背景进行赋值并可在线监视。

步骤：

1. 在SIMATIC Manager中，双击打开背景数据块。
2. 如果需要打开“Parameter Assignment for Data Blocks”(对数据块参数赋值)，点击“Yes”回答对话框弹出的提示。结果：打开背景DB。
3. 通过菜单命令**View > Data View** 或 **View > Declaration View**显示数据块不同视窗。对于带有“S7_techparam”系统属性的背景数据块或多重背景，自动打开“technological parameters”窗口。
4. 按需求编辑背景数据块。在消息窗口中显示相关的信息、警告信息或错误信息。双击相应的警告或错误信息可以进入相应位置。
5. 通过菜单命令**PLC > Download Parameter Setting Data**可以将修改的实际值从PG下载到CPU中。
6. 通过菜单命令**Debug > Monitor**显示打开块的程序状态，并在线监视所下载的实际值。

注意

可以识别带有“S7_techparam”系统属性的数据块。察看一个块是否具有该系统属性，可进入SIMATIC Manager并选择该块，通过菜单命令**Edit > Object Properties**并打开属性框。

12.1 对技术功能参数赋值

使用 “Parameter Assignment for Data Blocks” 功能，可以方便地对标准库提供的温度控制块FB 58 “TCONT_CP”和FB 59 “TCONT_S”进行参数赋值并可在线监视。

步骤：

1. 在SIMATIC Manager中，通过**File > Open > Libraries**选择并打开STEP 7标准库。
2. 选择“PID Control Blocks”然后点击“Blocks”。可以发现带“S7_techparam”属性的功能块：
   - FB 58 “TCONT_CP”：带连续或脉冲输入信号的温度控制器
   - FB 59 “TCONT_S”：积分型执行器的温度控制器
3. 将功能块(FB 58或FB 59)从标准库中复制到项目中。
4. 通过**Insert > S7 Block > Data Block**为所选的FB创建一个背景DB。
5. 在SIMATIC Manager中双击打开背景DB，并启动功能“Parameter Assignment for Data Blocks”。结果：在技术窗口中打开背景DB。现在可以轻松地对背景DB进行参数赋值并在线监视。
6. 在窗口中输入适当的控制值。在消息窗口中显示相关的信息、警告信息或错误信息。双击相应的警告或错误信息可以进入相应位置。

注意

可以识别带有“S7_techparam”系统属性的数据块。察看一个块是否具有该系统属性，可进入SIMATIC Manager并选择该块，通过菜单命令**Edit > Object Properties**并打开属性框。

13 生成STL源文件

13.1 编写 STL 源文件的基本信息

可以将用户程序或部分程序作为STL源文件，源文件可以包括多个程序块代码，源文件经过编译后可以生成所有包含的程序块。

使用源文件生成程序有如下优势：

- 用任意ASCII编辑器编写源文件，然后导入编译生成块，这些生成的块存贮在S7用户程序文件夹中。
- 可以在一个源文件中编写多个块。
- 即使含有语法错误，源文件也可以保存，如果在编辑逻辑块时错误，则不能保存。只要在源文件编译时才会出现语法错误提示。

用语句表（STL）语法编写源文件，使用关键字在源文件中设定块结构、声明变量表和程序网络段。

当生成STL源文件时，应注意以下几点：

- 编写STL源文件的有关注意。
- STL源文件中块的语句及格式。
- 块在STL源文件中的结构。

13.2 编写 STL 源文件的规则

13.2.1 在STL源文件输入语句的规则

STL源文件由文本组成，为将其编译成块，必须按一定规则编写其结构及语句。

生成用户程序作为STL源文件参考下列规则：

标　题	规　则
语句	除CALL指令结构以外，与STL语句表中的语法相同。
CALL	在源文件中将参数输入到括号内，并将各参数用逗号分开。 例如：FC call（一行指令） CALL FC10 （param1：=I0.0，param2：=I0.1）； 例如：FB call （一行指令） CALL FB10，　DB100 （para1：=I0.0，para2：=I0.1）； 例如：FB call （多行指令） CALL FB10，　DB100 （para1：=I0.0，　para2：=I0.1）； **注意：** 在ASCII码中调用块时，按定义的顺序完成参数的传送。否则STL与源文件中的赋值就会不匹配。
大/小写	除系统属性和跳转标签以外，编辑器不能识别大小写。当输入字符串时（数据类型STRING），必须确定大小写。 关键字以大写字母显示。在编译的时候看不到大小写，因此可以用大写、小写或大小写混合方式输入关键字。
分号	在STL语句和变量表的结尾使用分号（；）。这样就可以在一行输入多条语句。
双斜线（// ）	双斜线（// ）作为注释的开始，RETURN作为注释的结束。

13.2.2 在STL源文件中声明变量的规则

源文件中的每一个块都要声明所需的变量。

变量声明表应设置在每个块的代码部分之前。

如果某些变量正在被调用，应按声明类型以正确的顺序声明这些变量，也就是说一个声明类型的变量数据类型相同。

以梯形图、功能块图和语句表方式编程时，需要填写变量声明表，在这里需要设置相关的关键字。

变量声明表的关键字

声明类型	关键字	用于
输入参数	″VAR_INPUT″ 声明表 ″END_VAR″	FB，FC
输出参数	″VAR_OUTPUT″ 声明表″ END_VAR″	FB，FC
输入/输出参数	″VAR_IN_OUT″ 声明表″ END_VAR″	FB，FC
静态变量	″VAR″ 声明表 ″END_VAR″	FB
临时变量	″VAR_TEMP″ 声明表″ END_VAR″	OB，FB，FC

关键字END-VAR表示声明表结束。

声明表是以某声明类型组成的变量表，表中的变量都有一个预置的值（VAR_TEMP除外）。下面是一个声明表结构的例子：

Duration_Motor1：	S5TIME	：=	S5T#1H_30M；
Variable	Data type		Default value

注意：

- 变量符号必须以字母开始。符号不能与关键字相同。
- 如果局部变量声明表与符号表中的变量符号相同，则应在局部声明表中的变量名前增加“#”符号，在符号表中的变量应加上引号，否则，此变量视为局部变量。

13.2.3 STL源文件中块次序的规则

被调用的块应在调用它的块之前生成。这意味着：

- 在大多数情况下，OB1用于调用其它的块，所以被调用的块应在OB1之前生成，OB1应在最后生成。
- 用户定义的数据类型（UDT）应在使用它的块之前创建。
- 与用户定义的数据类型（UDT）相关联的数据块应在用户定义的数据类型之后创建。
- 如多个块使用同一数据块，则应首先创建这个数据块。
- 背景数据块应在其相应的功能块之后。
- DB0备用。用户不能建立DB0。

13.2.4 在STL源文件中设定系统属性的规则

在块和参数中可以设置系统属性。系统属性用于管理消息组态、连接组态、操作员接口功能及过程控制组态等功能。

在源文件中设置系统属性时有如下规则：

- 系统属性的关键字以“S7_”开始
- 系统属性应位于括号内。
- 语法格式：{S7_idenifier：= 'string'}
  使用多个标识符应用“；”分开。
- 用于块的系统属性应在块属性之前，在关键字ORGANIZATION_和TITLE之后。
- 参数的系统属性与参数声明在一起，并且位于数据声明冒号之前。
- 大小写字符不同，在输入时大小写应有所区别。

在“System Attributes”下面，使用菜单命令**File > Properties**，可以检查或修改用于块的“属性”标签。

使用菜单命令**File > Object Properties**，检查或修改用于参数的系统属性，光标必须放在参数声明的名称上。

13.2.5 在STL源文件中设定块属性的规则

块属性可使用户更容易辨识所生成的各程序块，并且还可以对这些程序块加以保护，防止非法修改。

在“General Part1”和“General Part2”标签下，使用菜单命令**File > Properties**，可以检查或修改软件块属性。

其它块属性可以依次在源文件中输入。

请按下列规则在源文件中输入：

- 块属性应在变量声明表之前
- 每个块属性各占一行
- 各行用分号结束
- 用关键字指定块属性
- 输入的块属性在块属性表中顺序显示
- 每个块类型有效的块属性列表：从块属性到块类型

注意：

块属性能够在SIMATIC Manager中显示并能在此编辑下列属性：AUTHOR、FAMILY、NAME和VERSION。

块属性和软件块次序

当输入块属性时，输入顺序如下表。

次序	关键字/属性	含义	举例
1.	[KNOW_HOW_ PROTECT]	块保护；使用该指令后，当该块进行编译时，使程序部分不可见。只能看到块的接口，但不能修改。	KNOW_HOW_PROTEC T
2.	[AUTHOR：]	作者名：公司名，部门名或其它名称（最多为没有空格的8个字符）	AUTHOR：Siemens，没有关键字
3.	[FAMILY：]	块系列名：例如，控制器（最多为没有空格的8个字符）	FAMILY：Controller，没有关键字
4.	[NAME：]	块名（最多8个字符）	NAME：PID，没有关键字
5.	[VERSION：int1.int2]	块的版本号（两个数的范围均在0到15之间，意为0.0到15.15）	VERSION：3.10
6.	[CODE_VERSION1]	功能块能否有多重背景的标识符。如果想用多重背景则功能块不应有该属性。	CODE_VERSION1
7.	[UNLINKED] 只适用于DB	具有“UNLINKED” 属性的数据块只能存储在装载存储器中,它不占用工作存储器的空间,也不能与程序连接。不能使用MC7的命令访问，只能调用SFC20 BLKMOV或SFC 83 READ_DBL将DB传送到工作存储器。	
8.	[READ_ONLY] 只适用于DB	数据块的写保护；其数据只能读，不能修改。	FAMILY=Examples VERSION=3.10 READ_ONLY

13.2.6 每个块类型的许可块属性

下表所示为哪种块类型可具有哪些块属性：

特性	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	•	•	•	•	-
AUTHOR	•	•	•	•	-
FAMILY	•	•	•	•	-
NAME	•	•	•	•	-
VERSION	•	•	•	•	-
UNLINKED	-	-	-	•	-
READ_ONLY	-	-	-	•	-

使用KNOW_HOW_PROTECT设定块保护

当用户在STL源文件中编程块时，通过使用关键字“KNOW_HOW_PROTECT”设定块保护，可以保护块，防止非授权用户使用。

该块保护命令具有下列效果：

- 如果需要在STL、FBD或梯形图编辑器中查看编译后的保护块，则该块的程序部分不能显示。
- 该块的变量声明表中只显示类型为 var_in、var_out和var_in_out的变量。类型为var_stat和var_temp的变量隐含。
- 在输入任何其它块属性之前，输入关键字KNOW_HOW_PROTECT。

使用READ_ONLY，设定数据块为写保护

对于数据块，用户可以设定写保护，以防在程序执行期间改写数据块。为此，数据块必须以STL源文件的形式存在。

使用源文件中的关键字“READ_ONLY”，可以设置写保护。这个关键字必须输入在声明变量之前的那一行。

13.3 STL 源文件中块的结构

在STL源文件中用关键字设定块结构。在下列块中，块的结构不同：

- 逻辑块
- 数据块
- 用户定义的数据类型（UDT）

13.3.1 STL源文件中块的结构

逻辑块由下列部分组成，各部分由相应的关键字定义：

- 块的开始部分
- 由关键字定义块的号码和名称，例如：
  - “ORGANIZATION_BLOCK OB1”定义组织块
  - “FUNCTION_BLOCK FB6”定义功能块，或
  - “FUNCTION FC1：INT”定义功能。同时设定了功能的类型。它可以是简单或复合的数据类型（ARRAY和STRUCT除外），并定义返回值的数据类型（RET_VAL）。如果没有返回值，则使用关键字“VOID”。
- 关键字“TITLE”引导可选择的块的标题（标题最长64个字符）。

- 用双斜线作为注释的开始
- 块的属性（可选）
- 变量声明表
- “BEGIN”作为程序部分的开始。程序部分由1个或多个程序段组成，仅用“NETWORK”作为程序段的标识符。不能输入程序段的号码。
- 关键字“TITLE=”作为程序段的开始。（标题最长64个字符）
- 用双斜线“// ”作为每个程序段注释的开始。
- END_ORGANIZATION_BLOCK，END_FUNCTION_BLOCK或END_FUNCTION作为块的结束。
- 块类型和号码之间应加空格。块的符号名应加上引号，以确保局部变量符号名与符号表中的名称一致。

13.3.2 在STL源文件中数据块的结构

数据块由下列部分组成，各部分由自己的关键字引导：

- 启动部分：关键字和块号码或块名称，例如：DATA_BLOCK DB26
- 参考UDT或功能块（可选）
- 由关键字“TITLE=”引导可选的块标题。（最长64个字符）
- 用双斜线“// ”作为块注释的开始
- 块的属性（可选）
- 变量声明表（可选）
- 带有缺省值的赋值部分以BEGIN开始（可选）
- END_DATA_BLOCK作为块的结束

数据块有三种类型：

- 用户定义的数据块
- 与用户定义的数据类型（UDT）关联的数据块
- 与功能块关联的数据块（作为背景数据块）

13.3.3 在STL源文件中用户定义数据类型的结构

用户定义的数据类型由下列部分组成，各部分由自己的关键字引导：

- 启动部分：关键字TYPE和块号码或块名称，例如：TYPE UDT20
- 结构化数据类型
- END_TYPE作为块的结束

当输入用户定义的数据类型时，数据类型应位于使用它的块之前。

13.4 在 STL 源文件中块的语句及格式

如果编写STL源文件，格式表将显示语法和格式。语法说明如下：

- 各部分的说明在右边一栏。
- 输入内容应用引号。
- 方括号[...]其中内容可选择填写。
- 用大写字符输入关键字。

13.4.1 组织块的格式表

结　构	说　明
"ORGANIZATION_BLOCK" ob_no 或ob_name	ob_no—块的符号，如OB1； ob_name—在符号表中定义的符号名
[TITLE=]	块标题（最长64个字符）
[块注释]	块注释应从“// ”开始
[块的系统属性]	块的系统属性
[块的属性]	块的属性
变量声明表	临时变量声明表
"BEGIN"	关键字用于变量声明表和STL指令部分之间
NETWORK	段的开始
[TITLE=]	段标题（最多64个字符）
[段注释]	块注释应从“// ”开始
STL指令部分	块指令
"END_ORGANIZATION_BLOCK"	组织块结束的关键字

13.4.2 功能块的格式表

下表是STL源文件中功能块的简单格式表：

结　构	说　明
"FUNCTION_BLOCK" fb_no 或 fb_name	fb_no ：符号表中的符号名，如FB6；fb_name 符号表中的符号名
[TITLE=]	块标题（最长64个字符）
[块注释]	块注释应从“// ”开始
[块的系统属性]	块的系统属性
[块的属性]	块的属性

结　构	说　明
变量声明表	（输入、输出及输入/输出参数），临时变量，静态变量的声明表参数声明表包括参数的系统属性声明
"BEGIN"	关键字用于变量声明表和STL指令部分之间
NETWORK	段的开始
[TITLE=]	段标题（最多64个字符）
[段注释]	块注释应从“// ”开始
STL指令部分	块指令
"END_FUNCTION_BLOCK	表示功能块结束的关键字

13.4.3　功能格式表

下表是STL源文件中功能的简单格式表：

结　构	说　明
“FUNCTION” fc_no：fc_type 或fc_name：fc_type	fc_no：为块编号，如FC5； fc_name为符号表中所定义的块符号名； fc_type：功能返回值（RET_VAL）的数据类型。它是一种简单或复合的数据类型（ARRAY和STRUCT除外）或VOID。 如使用系统属性作为返回值（RET_VAL），必须将用于参数的系统属性输入在数据声明表之前。
[TITLE=]	块标题（最长64个字符）
[块注释]	块注释应从“// ”开始
[块的系统属性]	块的系统属性
[块的属性]	块的属性
变量声明表	（输入、输出及输入/输出参数），临时变量，静态变量
"BEGIN"	关键字用于变量声明表和STL指令部分之间
NETWORK	段的开始
[TITLE=]	段标题（最多64个字符）
[段注释]	块注释应从“// ”开始
STL指令部分	块指令
“END_FUNCTION”	表示功能结束的关键字

13.4.4 数据块的格式表

下表是STL源文件中数据块的简单格式表：

结　构	说　明
"DATA_BLOCK"　db_no或 db_name	db_no：块的编号，如DB5； db_name：符号表中的符号名
[TITLE=]	块标题（最长64个字符）
[块注释]	块注释应从“// ”开始
[块的系统属性]	块的系统属性
[块的属性]	块的属性
声明部分	声明此块是否与UDT或FB相关，定义块的号码或符号表中的符号名或某种复合的数据
"BEGIN"	关键字用于变量声明表和STL指令之间
[初始赋值]	变量具有初始值各变量或是赋值与常数值或赋与其它块相关的值
"END_DATA_BLOCK"	块结束的关键字

13.5 生成 STL 源文件

13.5.1 生成STL源文件

源文件必须在S7程序下的源文件夹中生成。可以在SIMATIC　Manager或编辑器窗口生成源文件。

在SIMATIC Manager中生成源文件

1. 通过双击打开相应的“源文件”文件夹。
2. 要插入一个STL源文件，可以用菜单命令**Insert > S7 Softwore > STL Source File**。

在编辑器窗口生成源文件

1. 选择菜单命令**File > New**。
2. 在对话框中选择包含用户程序和块的同一个S7程序里的源文件文件夹。
3. 为新源文件输入一个名字。
4. 用“OK”确认。

该源文件在输入的名字后生成，并显示一个编辑窗口。

13.5.2 编辑S7源文件

对于一个被编辑的源文件，可以在其对象特性中设置编程语言和编辑器。这就确保了当打开源文件编辑时，启动了正确的编辑器和正确的编程语言。STEP 7标准软件支持在STL源文件中编程。

可选软件提供其它的编程语言。如果可选的软件已装入计算机，可以选择菜单命令插入源文件。

要编辑一个S7源文件可按如下进行：

1. 通过双击打开相应的“源文件”夹。
2. 按下述方法启动所需的编辑器：
   - 双击右半窗口中所需的源文件。
   - 选择右半窗口中所需的源文件，并选择菜单命令Edit > Open Object。

13.5.3 设定源代码文本的布局

为了提高源文件的可读性，选择菜单命令Options > Settings以及“Source Code”。可以指定源代码中各个单元的字体、字号和颜色。

13.5.4 将块模板插入STL源文件

用于STL源文件编程的块模板有组织块（OB）、功能块（FB）、功能（FC）、数据块（DB）、背景数据块、与用户定义的数据类型相关的数据块、用户定义的数据类型（UDT）。块模板使得在源文件中输入块更容易，并且可以观察语法和结构框架。

要插入一个块模板可按如下进行：

1. 激活需要插入块模板的源文件窗口。
2. 将光标放在源文件中需要在其后插入块模板的位置。
3. 选择下面的菜单命令之一 Insert>Block Template>OB/FB/FC/DB / Instance DB/DB Referencing UDT/UDT。

块模板被插入到光标位置后的文件中。

13.5.5 插入其它STL源文件的内容

可以将其它源文件的内容插入到STL源文件中。步骤如下：

1. 激活要插入其它源文件的目标源文件的窗口。
2. 将光标指向该目标文件中要插入的位置。
3. 选择菜单命令Insert > Object > File。
4. 在出现的对话框中选择要插入的源文件。

所选择的源文件内容插入到光标位置之后。

13.5.6 从STL源文件现有的块中插入源代码

可以将其它块的源代码插入到STL源文件中，该源代码可以是用LAD、FBD或STL编写的。它可以是OB、FB、FC、FB、UDT。其步骤如下：

1. 激活要插入块的目标源文件的窗口。
2. 将光标指向该块中要插入的位置。
3. 选择菜单命令Insert > Object > Block。
4. 在出现的对话框中选择要插入的块。

13.5.7 插入外部源文件

可用ASCII编辑器生成并编辑源文件，然后将它导入到项目中，再用这个应用程序将它编译成块。要实现这一步，必须将源文件导入到S7程序的“源文件”夹中，在编译过程中生成的块将存储在该S7程序的S7用户程序中。

要插入外部源文件可按如下进行：

1. 选择S7程序的源文件夹，外部源文件将被导入到该文件夹中。
2. 选择菜单命令Insert > External Source File。
3. 在显示的对话框中选中要导入的源文件。

要导入的源文件必须有有效的扩展名。STEP 7使用扩展名来判定源文件的类型。这就意味着，当STEP 7导入扩展名为.AWL的文件时建立STL源文件。有效的文件扩展名在对话框的“File Type（文件类型）”中列出。

注意

也可以用菜单命令Insert > External Source File，导入用STEP 7版本1生成的源文件。

13.5.8 从块建立STL源文件

可以用已有的块生成一个可以在任何文本编辑器中编辑的STL源文件。该源文件生成在S7用户程序的源文件夹中。

要用块生成一个源文件可按如下进行：

1. 在程序编辑器中，选择菜单命令File > Generate Source File。
2. 在对话框中选择源文件夹。
3. 在文本框中为源文件输入名字。
4. 在“选择STEP 7块”对话框中，选择想用来生成特定的源文件的块。所选的块显示在右边的列表框中。
5. 用“OK”确认。

来自所选块的STL源文件被生成并显示在编辑窗口中。

13.5.9 导入源文件

可以将源文件从任何目录中导入到项目中：

1. 在SIMATICManager中，选择要导入到的文件夹。
2. 选择菜单命令Insert > External Source File。
3. 在显示的对话框中，选择要插入的源文件。
4. 点击“Open”。

13.5.10 导出源文件

可以将源文件从项目中导出到任何目录中：

1. 在源文件夹中选择源文件。
2. 选择菜单命令Insert > Export Source File。
3. 在显示的对话框中，选择目的路径和文件名。
4. 点击“Save”。

注意：

如果对象没有文件扩展名，将对该文件加入该文件类型的扩展名。例如：STL源文件“prog”导出的文件为“prog.awl”。

如果该对象已经有了一个合法的扩展名，则将保持该文件扩展名。

如果该对象的扩展名不合法(指文件名中含有点)，则不会对其加扩展名。

“File type”下的“Export Source File”对话框中列出了可以使用的合法的扩展名。

13.6 存储并编译 STL 源文件并执行一致性检查

13.6.1 存储STL源文件

可以在任何时候在一个STL源文件的当前状态下对其进行存储。程序并不编译也不运行语法检查，这意味着任意错误都可以同时被存储。

语法错误的检查和报告只在源文件编译时或一致性检查后进行。

要在同一名下存储源文件

1. 激活要存储的源文件的窗口。
2. 选择菜单命令File > Save。

要将源文件存在一个新名/另一个项目下

1. 激活要存储的源文件的窗口。
2. 选择菜单命令File > Save As。
3. 在对话框中选择要在其中存储源文件的文件夹并输入新名字。

13.6.2 检查STL源文件的一致性

用菜单命令File > Consistency Check，可以显示源文件中的所有语法错误。与编译相比，不会有块生成。

当一致性检查完成后，有对话框显示已发现的错误的总数。

所有发现的错误分别列在窗口的下部并各有一行参考。为了能够生成各个块，在编译前修改这些错误。

13.6.3 在STL源文件中诊断故障

激活的源文件窗口被分割为两部分。在下半部分显示以下错误：

- 用菜单命令File > Compile，启动编译运行后所发现的错误。
- 用菜单命令File > Consistency Check，启动一致性检查后所发现的错误。

要找到错误在源文件中的位置，将光标放在窗口下部相应的错误信息上。窗口上部包含错误的文件行自动加亮。错误信息也显示在状态栏中。

13.6.4 编译STL源文件

要求

为了将你用STL源文件生成的程序编译成块，必须满足下列要求：

- 只有存储在S7程序下的“源文件”夹中的源文件才能被编译。
- 除了“源文件”夹之外，还要有一个“块”文件夹在S7程序下，编译过程中生成的块将存储在这个“块”文件中。只有源文件编译不出错。源文件中编程的块才能生成。如果在一个源文件中有许多块，只有那些没有错误的块才能生成。然后你就可以打开这些块编辑它们，下载CPU并一个一个地进行调试。

编辑器源文件

1. 打开要编译的源文件。该源文件必须在S7程序的源文件夹中，编译生成的块也要存储在这个S7程序的S7用户程序中。
2. 选择菜单命令File > Compile。
3. “编译报告”对话框显示出来，显示已编译的行和已发现的错误。

只有当源文件编译无错时才能生成文件指定的块。如果在一个源文件中有许多块，只有那些没有错误的块才能生成。错误警告不妨碍块的生成。

编译过程中查到的所有错误都显示在工作窗口的下部，并且在生成各个块之前必须进行改正。

在SIMATIC Manager中的编译过程

1. 通过双击打开合适的“源文件”夹。
2. 选择一个或多个要编译的源文件。对于一个关闭的源文件夹，无法编译其包含的源文件。
3. 选择菜单命令File > Compile，开始编译。为源文件选择的编译器被调用。被成功编译的块则存储在S7程序的块文件夹中。 编译过程中查出的所有语法错误显示在对话框中，并且必须被纠正，否则在块中生成错误。

13.7 STL 源文件的例子

13.7.1 STL源文件声明变量的例子

基本类型的变量

```
                          // 注释与声明表双斜线分开。
VAR_INPUT                 // 关键字：输入变量
in1：INT；                // 用“：”将变量名和类型分开
```

```
in3：DWORD；                    // 变量声明用分号结束
in2：INT：= 10；                // 设置声明表中的初始值
END_VAR                         // 同一类型的变量声明结束
VAR_OUTPUT                      // 关键字：输出变量
out1WORD；
END_VAR                         // 关键字：临时变量
VAR_TEMP
temp1：INT；
END_VAR
```

数组型变量

```
VAR_INPUT                       // 输入变量
array1：ARRA [1..20] of INT；                 // 数组1是1维数组
array2：ARRAY [1..20，1..40] of DWORD；       // 数组2是2维数组
END_VAR
```

结构体变量

```
VAR_OUT                         // 输出变量
OUTPUT1：STRUCT                 // OUTPUT1具有STRUCT数据类型
var1：BOOL；                    // 结构体中的元素1
var2：DWORD；                   // 结构体中的元素2
END_STRUCT；                    // 结构体结束
END_VAR
```

13.7.2 STL源文件中组织块例子

```
ORGANIZATION_BLOCK OB1
TITLE = Example for OB1 with different block calls
// 第3段显示块的调用
// 带有或不带参数

{S7_pdiag：= 'true'}            // 块的系统属性
AUTHOR          Siemens
FAMILY          Example
NAME            Test_OB
VERSION         1.1
VAR_TEMP
Interim value：INT；            // 缓存器
END_VAR
```

```
BEGIN

NETWORK
TITLE = Function call transferring parameters
//  传送参数只有1行
CALL FC1 （param1：=I0.0，param2：=I0.1）；

NETWORK
TITLE = Function block call
// 传送参数
// 传送参数多行
CALL Traffic light control，DB6（                // FB名称，背景数据块
dur_g_p            ：= S5T#10S，                 // 参数赋实际值
del_r_p            ：= S5T#30S，
starter            ：= TRUE，
t_dur_y_car        ：= T 2，
t_dur_g_ped        ：= T 3，
t_delay_y_car      ：= T 4，
t_dur_r_car        ：= T 5，
t_next_red_car     ：= T 6，
r_car              ：= "re_main"，               // 引号表示符号
y_car              ：= "ye_main"，               // 在符号表输入名称
g_car              ：= "gr_main"，
r_ped              ：= "re_int"，
g_ped              ：= "gr_int"）；

NETWORK
TITLE = Function block call
// 传送参数
// 传送参数1行
CALL FB10， DB100（para1：=I0.0，para2：=I0.1）；
END_ORGANIZATION_BLOCK
```

13.7.3 STL源文件中功能的例子

```
FUNCTION FC1：VOID
//  仅用于调用
VAR_INPUT
   param1：bool；
   param2：bool；
END_VAR
begin
end_function
```

```
FUNCTION FC2：INT
TITLE = Increment number of items
// 传送条目<1000个，这个功能增加了传送条目个数。
// 如果传送条目数超出1000个，
// 则在返回值功能（RET_VAL）显示“-1”。

AUTHOR          Siemens
FAMILY          Throughput check
NAME：          INCR_ITEM_NOS
VERSION：       1.0

VAR_IN_OUT
ITEM_NOS：INT；                    // 当前制造零件号
END_VAR

BEGIN

NETWORK
TITLE = Increment number of items by 1
// 运行的条目数少于1000个
// 计数器加1
L ITEM_NOS；  L 1000；                       // 多于1个的例子
> I；  JC ERR；                              // 行指令
L 0；  T RET_VAL；
L ITEM_NOS；  INC 1；  T ITEM_NOS；  BEU；
ERR：L -1；
T RET_VAL；
END_FUNCTION

FUNCTION FC3 {S7_pdiag：= ’true’}：INT
TITLE = Increment number of items
// 传送条目<1000个，这个功能
// 增加了传送条目个数。如果传送条目数超出1000个，
// 则在返回值功能（RET_VAL）显示“-1”。
// RET_VAL具有参数的系统属性

AUTHOR：        Siemens
FAMILY：        Throughput check
NAME：          INCR_ITEM_NOS
VERSION：       1.0
```

```
VAR_IN_OUT
ITEM_NOS {S7_visible：= 'true'}：INT；      // 当前制造零件号
// 参数的系统属性
END_VAR

BEGIN

NETWORK
TITLE = Increment number of items by 1
// 运行的条目数少于1000个
// 计数器加1
L ITEM_NOS；  L 1000；                      // 多于1个的例子
I；  JC ERR；                               // 行指令
L 0；  T RET_VAL；
L ITEM_NOS；INC 1；T ITEM_NOS；BEU；
ERR：L -1；
T RET_VAL；

END_FUNCTION
```

13.7.4 STL源文件中功能块例子

```
FUNCTION_BLOCK FB6
TITLE = Simple traffic light switching
// 行人穿越
// 主路时指示灯的控制

{S7_m_c：= 'true'}          // 块的系统属性
AUTHOR        ：            Siemens
FAMILY        ：            Traffic light
NAME          ：            Traffic light01
VERSION       ：            1.3

VAR_INPUT

Starter     ：   BOOL：= FALSE；  //  对行人的要求
t_dur_y_car：   TIMER；           // 绿灯亮时对行人的要求
t_next_r_car：  TIMER；           // 红灯亮时对行人的要求
t_dur_r_car ：  TIMER；
```

```
number      {S7_server：= 'alarm_archiv'；  S7_a_type：= 'alarm_8'}：
DWORD；
// 车的号码
// 号码是具有系统属性的参数

END_VAR
VAR_OUTPUT
g_car：      BOOL：   =FALSE；       //  绿灯对车的要求

END_VAR
VAR
condition：  BOOL：   =FALSE；       //  红灯对车的出现的条件
END_VAR

BEGIN
NETWORK
TITLE = Condition red for main street traffic
// 1分钟后，行人穿越主路的绿灯
// 作为红灯亮条件
// 主路交通量
            A（；
            A         #starter；            // 行人进路时绿灯的要求
            A         #t_next_r_car；           // 红灯亮的时间
            O         #condition；          // 或红灯亮的条件
            ）；
            AN        #t_dur_y_car；        // 红灯不亮的条件
            =         #condition；          // 红灯亮的条件

NETWORK
TITLE = Green light for main street traffic
            AN        #condition；          // 主街上没有红灯亮的条件
            =         #g_car；              //  主街上的绿灯

NETWORK
TITLE = Duration of yellow phase for cars
                 // 增加控制交通灯的程序
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB10
VAR_INPUT
  para1：bool；
  para2：bool；
end_var
```

```
begin
end_function_block

data_block db10
FB10
begin
end_data_block

data_block db6
FB6
begin
end_data_block
```

13.7.5 STL源文件中的数据块举例

数据块

```
DATA_BLOCK DB10
TITLE = DB Example 10
STRUCT
        aa：BOOL；    // Bool类型的变量
        bb：INT；         // INT类型的变量INT
        cc：WORD；
END_STRUCT；

BEGIN    // 赋初值
            aa：= TRUE；
            bb：= 1500；
END_DATA_BLOCK
```

与用户定义的数据类型关联的数据块

```
DATA_BLOCK DB20
TITLE = DB （UDT） Example
UDT 20                  // 用户定义的相关数据类型
BEGIN
            start：= TRUE；    // 赋初值
            setp.：= 10；
END_DATA_BLOCK
```

注意：

所用的UDT必须在源文件中数据块的前面。

功能块背景数据块：

```
DATA_BLOCK DB30
TITLE = DB （FB） Example
FB30                                    // 相关功能块
BEGIN
                start：= TRUE；      // 赋初值
                setp：= 10；
END_DATA_BLOCK
```

注意：

相关功能块必须在源文件中的数据块之前。

13.7.6 STL源文件中用户定义数据类型的举例

```
Type  UDT20
STRUCT
            Start：BOOL；        // Bool类型的变量
            Setp： INT；          // INT类型的变量
            数值：WORD；         // WORD类型的变量
END_STRUCT；
END_TYPE
```

14　显示参考数据

14.1　可用参考数据概述

生成并评估参考数据可使用户程序的调试和修改更加容易。参考数据具有以下作用：

- 作为整个用户程序的一个概述
- 作为修改和测试的基础
- 作为程序文档的补充

下表说明在各个视窗中可以摘取的信息：

视窗	目的
交叉参考表	在存储区域I、Q、M、P、T、C 以及DB、FB、FC、SFB、SFC块中由用户程序使用的地址概述。 用菜单命令View > Cross Reference for Address，可以显示包括对地址多次访问在内的所有交叉参考数据。
输入、输出及位存储赋值表	用户程序中已使用的定时器和计数器（T和C）以及I、Q、M存储区中的位地址的概述；为故障诊断或修改用户程序奠定了重要基础。
程序结构	在一个用户程序内块的分层调用结构，以及所使用的块及其嵌套层次的概述。
未用符号	所有已在符号表中定义但未在用户程序中使用的符号概述，这些用户程序有可供使用的参考数据。
无符号地址	在用户程序中使用的绝对地址，已经产生参考数据但是没有在符号表中定义。

用户程序的参考数据包括表中所列各项。可以为一个用户程序或多个用户程序生成并显示一种或多种列表。

同时显示多个视窗

可以在附加窗口显示其它列表，例如：

- 比较不同的S7用户程序的同一种列表。
- 显示某个列表的不同视窗，例如，不同显示的交叉参考表在屏幕上并行排列。还可以在一个交叉参考表中只显示S7用户程序的输入而在另一个列表中只显示输出。
- 同时打开一个S7用户程序的多个列表，例如，程序结构和交叉参考表。

14.1.1　交叉参考表

交叉参考表给出了S7用户程序所用地址的概述。

当显示交叉参考表时就可以得到输入（I）、输出（Q）、位存储器（M），定时器（T）、

计数器（C）、功能块（FB）、功能（FC）、系统功能块（SFB）、系统功能（SFC）、I/O（P）和数据块（DB）这些存储区域中被S7用户程序使用的地址列表，在该列表中显示它们的地址（绝对地址或符号地址）及使用情况。交叉参考表显示在一个激活的窗口中，在窗口的标题栏显示该交叉参考表所属的用户程序名。

窗口中的每一行都对应着交叉参考表的一个输入项。查寻功能可以很容易地找到指定的地址和符号。

当显示参考数据时，交叉参考表为缺省视窗。可以修改这个缺省设置。

结构

一个交叉参考表的输入项包括以下各栏：

栏	内容 / 含义
Address	绝对地址
Block	使用该地址的块
Type	对有关地址的访问是读（R）和 / 或写（W）
Language/Details	用于生成块的编程语言信息

只有为交叉参考表选择了相应的选项才会显示符号（Symbol）、块（Block）、类型（Type）和语言/明细数据（Language/Details）。块信息则依据该块编写时所用的编程语而定。

可以使用鼠标按照需要在屏幕上显示的交叉参考表中设置栏宽。

分类

交叉参考表的缺省选项是按存储区域分类。如果用鼠标点中栏标题，则可以按缺省分类标准对这一栏的输入项进行分类。

交叉参考表格式示例

地址	符号	块	类型	语言	详细数据
I1.0	Motor on	OB2	R	STL	Nw 2 Inst 33 /0
M1.2	MemoryBit	FC2	R	LAD	Nw 33
C2	Counter2	FB2		FBD	Nw2

14.1.2 程序结构

程序结构描述了在一个S7用户程序内块的分层调用结构。还可以对所用的块、它们的从属关系以及它们对局域数据的需求有一个概括的了解。

在“Generating Reference Data（生成参考数据）”窗口，用菜单命令**View > Filter**打开一个标签对话框。在“Program Structure（程序结构）”标签中，可以设置程序结构如何显示。

可以选其中之一：

- 树形结构和
- 从属结构

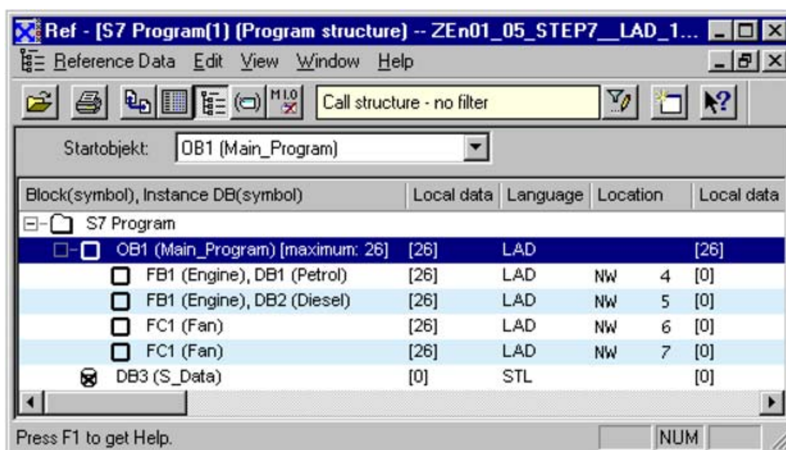
程序结构的符号

符号	含义
	正常调用的块（CALL FB10）
	无条件调用的块（UC FB10）
	条件调用的块（CC FB10）
	数据块
	递归调用
	递归且条件调用
	递归且无条件调用
	块未调用

- 递归调用在树形结构中被识别并以图形标记。
- 在分层结构中的递归调用被不同的符号标记。
- 正常调用的块（CALL），条件调用的块（CC）或无条件调用的块（UC）被标记为不同的符号。
- 未被调用的块显示在树形结构的底部并且标记有黑叉。对于未被调用的块不会有更进一步的调用分解。

调用结构

整个调用分层结构显示如下。



如果生成的程序结构中所有的组织块（OB）和OB1不在S7用户程序中，或者指定的起始块不在程序中，则会被自动提示指定另一个块作为程序结构的根。

对于显示块的多重调用，可以通过选项设置使之无效，这适用于调用结构，也适用于从属结构。

在调用结构中显示对局域数据的最大需求

为快速而概括了解用户程序中组织块对局域数据的需求，在树形结构中可显示以下内容：

- 每个OB所要求的最大局域数据数以及
- 每个路经所要求的局域数据

可以在“Program Structure”标签中激活或取消显示。

如果出现同步错误OB块（OB121，OB122），则在所要求最大局域数据的数字后面显示一个加号以及该同步错误OB块额外需求的局域数据量。

从属结构

从属结构显示项目中每个块和其它块之间的从属关系。

显示删除块

与删除块有关的行将显示为红色。

14.1.3 赋值表

赋值表将显示在用户程序中已经赋值的地址，对于用户程序的故障查寻和修改有很好的帮助作用。

I/Q/M赋值表的显示可以概括地了解输入(I)、输出(Q)和位存储器(M)、定时器(T)、计数器(C)中哪个字节的哪一位被使用了。I/Q/M赋值表在同一个工作窗口中显示。

这个工作窗口的标题栏显示该赋值表所属的S7用户程序名。

I/Q/M表

每一行包含存储区的一个字节，在该字节中有八个位，按其访问的情况标有代码。它还指示这个访问是否为一个字节、一个字或一个双字。

I/Q/M赋值表中的代码

白色背景	地址未被访问，因而也没有赋值
X	直接访问的地址
蓝色背景	间接访问的地址（字节、字或双字访问）

I/Q/M赋值表中每列含义

列	内容 / 含义
7 6 5 4 3 2 1 0	相应字节的位号
B	该字节以单字节访问的形式被占用
W	该字节以单字访问的形式被占用
D	该字节以双字访问的形式被占用

I/Q/M赋值表举例

下列例子显示了输入、输出和位存储（I/Q/M）的典型格式。

△	7	6	5	4	3	2	1	0	B	W	D
IB0		X	X	X	X	X	X				
IB1		X	X	X		X	X	X			
QB4						X	X	X			
QB5		X	X	X		X	X	X			
MB1											
MB2											
MB3											
MB4											
MB5											

第一行显示输入字节IB0的分配。输入地址IB0可直接访问（位访问），第1、2、3、4、5、6列识别为位访问“X”。

存储字节1和2、2和3或4和5为字访问，此时在W列上显示了一个数条并以蓝色背景显示。黑点表示字的起始字节。

T/C赋值表

每行显示10个定时器或计数器。

举例

	0	1	2	3	4	5	6	7	8	9
T00-09	•	T1	•	•	•		T6	•	•	•
T10-19	•	•	T12	•	•	•	•	T17	•	T19
T20-29	•	•	•	•	T24	•	•	•	•	•
C00-09	•	•	C2	•	•	•	•	C7	•	•
	0	1	2	3	4	5	6	7	8	9
C10-19	•	•	•	•	•	•	•	•	•	C19
C20-29	•	•	•	•	•	•	•	•	•	•
C30-39	•	•	•	•	C34	•	•	•	•	•

示例中定时器T1、T6、T12、T17、T19、T24及计数器C2、C7、C19、C34被程序中使用。

列表按字母排序，点击每列的标题可以从新排列次序。

14.1.4 未使用的符号

可以看到具有下列特征符号的一个概述显示：

- 已在符号表中定义的符号。
- 有参考数据的存在却未在用户程序的任何一部分中使用的符号。

它们在一个激活的窗口中显示。工作窗口的标题栏显示该列表所属的用户程序名。

窗口中显示的每行对应一个列表的条目。由地址，符号、数据类型和注释组成一行。

行	内容/含义
地址	绝对地址
数据类型	地址的数据类型
备注	符号表中地址备注

未用符号的格式举例

Symbol	Address	Data Type	Comment
MCB1	I 103.6	BOOL	Motor circuit breaker 1
MCB2	I 120.5	BOOL	Motor circuit breaker 2
MCB3	I 121.3	BOOL	Motor circuit breaker 3

可以通过点中栏标题来对条目分类，也可以在列表中将不再使用的符号删除，如果需要删除，在列表中选择符号然后执行“Delete symbols”功能。

14.1.5 没有符号的地址

当显示没有符号的地址列表时，可以得到在S7用户程序中使用了但未在符号表中定义的地址的列表。它们在一个激活的窗口中显示。这个工作窗口的标题栏显示该列表所属的用户程序名。

每一行包括地址和该地址在用户程序中使用的次数。

例如：

地址	次数
Q 2.5	4
I 23.6	3
M 34.1	20

也可以对没有符号的地址分配名称，如需这样，在列表中选择地址然后执行“Edit symbols”功能。

14.1.6 显示LAD、FBD和STL的块信息

在交叉参考列表和程序结构中可以显示梯形图、功能块图和语句表的编程语言信息。信息包括块的语言和明细数据。

如果在“Program Structure”(程序结构)中将过滤器设置为“Call Structure”(调用结构)并且选择了各自的选项，则程序结构显示相关的语言信息。

通过菜单命令**View > Filter**可以显示或隐藏“交叉参考”中的语言信息。

- 在“Filter（筛选器）”对话框“cross Reference（交叉参考）”标签中激活“Block language and details（块语言和明细数据）”复选框以显示语言信息。

语言信息根据块编写时的编程语言的不同而变化，用缩写显示。

语言	段	语句	指令
STL	Nw	Inst	/
LAD	Nw		
FBD	Nw		

Nw和**Inst**指示在哪个段和哪条语句中使用了该地址（交叉参考列表）或调用了该块（程序结构）。

为可选编程语言显示块信息

如果安装了相应的可选软件包则可访问有关块信息的在线帮助。

14.2 使用参考数据

14.2.1 参考数据显示方式

以下是可用来显示参考数据的方法：

从SIMATICManager中显示

1. 在离线项目窗口中，选择“Blocks（块）”文件夹。
2. 选择菜单命令**Options > Reference Data > Display**。

从编辑器窗口显示

1. 在“Blocks”文件夹中打开一个块。
2. 在块编辑器窗口中，选择菜单命**Options > Reference Data**。

“Customize（用户自定义）”对话框将显示，在此，用户可以首先选择所显示的窗口。缺省窗口为应用程序中显示参考数据最后关闭的窗口。用户可以取消对话框，以便以后调用。

从编译的块直接显示

可以从语言编辑器中的一个编译的块直接显示参考数据，以获得用户程序的当前概况。

14.2.2 在附加的工作窗口显示列表

使用菜单命令Window > New Window，可以打开另一个工作窗口，显示参考数据的其它视窗（例如，未使用的符号列表）。

使用菜单命令Reference Data > Open，可以为以前隐藏的参考数据打开一个工作窗口。

选择“View（视窗）”菜单中的某个命令或工具栏中相应的按钮，可以切换到其它的参考数据视窗：

参考数据视窗	显示该参考数据视窗的菜单命令
没有符号的地址	View > Addresses Without Symbols
未使用的符号	View > Unused Symbols
赋值	View > Assignment
程序结构	View > Program Structure
交叉参考列表	View > Cross References

14.2.3 生成并显示参考数据

生成参考数据：

1. 在SIMATICManager中，选择生成参考数据的块文件夹。
2. 在SIMATICManager中选择菜单命令Options > Reference Data > Generate。

在生成参考数据之前，计算机检查是否已有参考数据，如果有，是否为当前数据。

- 如果有参考数据，则已被生成。
- 如果已有参考数据不是当前的，可以选择是否更新这些参考数据或是否重新全部生成。

显示参考数据：

使用菜单命令Options > Reference Data > Display，可以显示参考数据。

在显示参考数据前，要作一个检查以确定是否已有参考数据存在，以及存在的参考数据是否是当前的。

- 如果没有参考数据存在则生成参考数据。
- 如果有不完整的参考数据存在，显示的对话框会提示该参考数据不一致。然后可以选择要更新该参考数据并且到什么程度。有以下几种可能：

选　择	含　意
只针对所修改的块	更新所修改的或新块的参考数据；任何已被删除块的信息将从参考数据库中删除
针对全部块	为所有块重新生成参考数据
不更新	不更新参考数据

为更新参考数据，块将再次被编译。调用适当的编译器以编译每一个块。使用菜单命令View > Update，可以更新在激活窗口中参考数据的视窗。

14.2.4　在程序中快速查找地址的位置

编程时使用参考数据，可使用光标查询某一个地址在程序中不同位置。如需这样，必须有最新的参考数据，但是，不必启动应用程序来显示参考数据。

基本步骤

1. 在SIMATICManager中，选择菜单命令Options > Reference Data > Generate，生成当前的参考数据。只有当没有参考数据时或参考数据是旧的，这一步才有必要。
2. 在一个打开的块中选择地址。
3. 选择菜单命令Edit > Go To > Instance。
   显示一个对话框，包含程序中所有该地址背景的列表。
4. 如果需要显示与被调用地址的物理地址或地址区域相重叠的那些地址背景，选择选项“Overlapping access to memory areas（地址区域的重叠访问）”，“地址”栏被加到表中。
5. 在列表中选中某位置并点击“Go To”按钮。

如果打开对话框时参考数据不是最新的，则会出现相关的信息，然后可以更新参考数据。

位置列表

对话框中的位置列表包含以下详细数据。

- 使用该地址的块。
- 块的符号名，如果该块有符号名的话。
- 详细数据，例如，位置信息和依据块或源文件（SCL）最初所用编程语言的指令。
- 依语言而定的信息。
- 对该地址的访问类型：只读（R）、只写（W）、读且写（RW）、未知（?）。
- 块语言。

可以对位置的显示作筛选，比如，只显示对某一个地址的写访问。关于此对话框，在线帮助提供了更多的关于在该区域中输入什么以及显示其它信息的详细信息。

注意

参考数据只能离线存在。因此该功能总是使用离线块的交叉参考。即使在一个在线块中调用该功能也是这样。

14.2.5 使用地址位置表的示例

要判定输出Q1.0（直接/间接）在哪个位置被置位了。以使用STL语言编写的OB1作为一个示例：

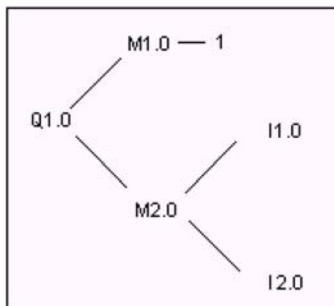
```
Network 1：  ...........
A Q1.0          // 在本例中
= Q1.1          // 无关

Network 2：
A M1.0
A M2.0
=Q 1.0          // 赋值

Network 3：
// 注释行
SET
=M1.0           // 赋值

Network 4：
A   I   1.0
A   I   2.0
=M2.0           // 赋值
```

得到以下的Q1.0的赋值关系树：



然后按如下进行：

1. 在LAD/STL/FBD编辑器中将光标定位于OB1的Q1.0（NW1，Inst1）上。
2. 选择菜单命令**Edit > Go To > Location**或用鼠标右键选择“Go to Location”。

   弹出的对话框显示Q1.0所有的赋值关系：

   OB1 Cycle Execution NW 1 Inst 3 /= W STL

   OB1 Cycle Execution NW 2 Inst 4 /A R STL
3. 在对话框中用“GO TO”按钮，跳到编辑器中的“NW2 Inst3（第二段第三条指令）”：
   Network 2：
   A M1.0
   A M2.0
   = Q 1.0
4. 现在必须检查对M1.0和M2.0的赋值。首先将光标定位于LAD/STL/FBD/编辑器中的M1.0上。
5. 选择菜单命令Edit > Go To > Location或用鼠标右键选择“Go to Location”。在弹出的对话框中显示M1.0的所有赋值关系：

   OB1 Cycle Execution NW 3 Inst 2 /= W STL

   OB1 Cycle Execution NW 2 Inst 1 /A R STL
6. 用对话框中的“Go To”按钮跳到编辑器中的“NW3 Inst2（第三段第二条指令）”。
7. 在LAD/STL/FBD编辑器中的第三段中可以看到，对M1.0的赋值不重要（因为总是1），因此应该检查M2.0的赋值。

   在早于V5的STEP 7版本中，必须重新把整个赋值顺序完整地检查一遍。按钮“>>”及“<<”能使这个操作简单一些。
8. 将打开的对话框“Go to Location”拖至前台，或从LAD/STL/FBD编辑器中在当前位置上调用功能“Go to Location”。
9. 点击“<<”按钮一次或两次直至显示Q1.0的位置；选择最后一个跳转位置“NW2 Inst 3”。
10. 用“GO TO”按钮（如第三点中所示）从地址位置对话框跳到编辑器中的“NWl Inst3”。
    Network 2：
    A M1.0
    A M2.0
    = Q 1.0
11. 在第4点中，检查了M1.0的赋值。现在要检查所有（直接/间接）对M2.0的赋值。将光标放在编辑器中M2.0上并调用功能“Go to Location：”所有对M2.0的赋值都显示出来：
    OB1 Cycle Execution NW 4 Inst 3 /= W STL
    OB1 Cycle Execution NW 2 Inst 2 /A R STL
12. 用“Go To”按钮跳到LAD/STL/FBD编辑器中的“NW4 Inst3”：
    Network 4：
    A I 1.0

A I 2.0
= M2.0

13. 现在要检查对I1.0和I2.0的赋值。在本例中不再描述这一过程，因为可以按照和以前一样的方式进行（从第4点开始）。

通过在LAD/STL/FBD编辑器和地址位置对话框之间进行切换，可以在用户程序中发现并检查地址相关的位置。

15 检查块的一致性和作为块属性的时间标签

15.1 检查块的一致性

简介

如果必须改编或扩展单个块的接口或代码，会导致时间标签冲突。同样，时间标签冲突会造成调用块和被调块或参考块之间的不一致，从而带来繁重的修正工作。

“Check block consistency”功能可以避免这种修正工作。“Check block consistency”功能可以清除大部分时间标签冲突和块的不一致性。对于不一致性不能自动消除的块，该功能可以切换到相应的编辑器中，在此，可以进行修改。所有块的不一致性都可以消除，并一步一步地进行编译。

要求

只有STEP 7 V5.0 Service Pack 3生成的项目，才能检查块的不一致性。对于较早的项目，在进行块的一致性检查时，必须首先编译每一项（菜单命令**Program > Compile All**）。

对于使用可选软件包生成的块，必须安装可选的软件包，才能检查块的一致性。

启动块的一致性检查

在开始进行块的一致性检查时，应检查块接口的时间标签，并且会造成不一致性的块将高亮显示在树状视窗中（从属树/调用树））。

1. 在SIMATIC管理器中，进入项目窗口，选择所需块的文件夹，通过菜单命令**Edit > Check Block Consistency**执行块的一致性检查。
2. 选择菜单命令**Program > Compile**。STEP 7可以自动识别相关块的编程语言，并调用相应的编辑器。应尽可能地自动修正时间标签冲突和块的不一致性，并编译块。如果不能自动清除块中的时间标签冲突或不一致性，在输出窗口中将出现错误信息（参见第3步）。对于树状视窗中的所有块，将自动重复这一过程。
3. 如果在编译运行时，不能自动清除所有块的不一致性，在输出窗口中，相应的块将标记为错误信息。将鼠标放在相应的错误处，并使用鼠标右键，调用弹出菜单中的错误显示。相关错误被打开，程序将跳到需要修改的位置。清除所有块的不一致性，保存并关闭块。对于所有标记为错误的块，重复这一过程。
4. 重新执行第2步和第3步。重复该过程，直至在信息窗口中不再有错误显示。

15.2 作为块属性的时间标签及时间标签冲突

块中包含一个代码时间标签和一个接口时间标签。这些时间标签可以被显示在块属性的对话框中。使用时间标签可以监控STEP 7程序的一致性。

STEP 7在进行时间标签比较时，如发现违背了规则，就会显示时间标签冲突。以下是可能出现违背规则的几种情形：

- 一个被调用的块比调用它的块的时间标签新（CALL）。
- 一个被参考的块比使用它的块的时间标签更新。

第二种违背规则的示例：

- 一个UDT比使用它的块的时间标签更新；这些块可以是一个DB或其它的UDT，或者在变量声明表中使用了该UDT的FC、FB、OB。
- 一个FB比其相应的背景数据块的时间标签更新。
- 一个FB2在FB1中被定义为多重背景，并且FB2的时间标签比FB1的更新。

注意

既使接口时间标记之间的关系是正确的，不一致性仍可能出现：

- 被参考块接口的定义与在它被使用的区域中的定义不匹配。

这些不一致就是所说的接口冲突。它们可能会出现在，比如，当块由不同的程序中拷贝出来或者当一个ASCII源文件编译时，不是程序中所有的块都生成了。

15.3 逻辑块中的时间标签

代码时间标签

块生成时的时间和日期就存储在这里。该时间标签会被刷新：

- 当程序代码被修改
- 当接口描述被修改
- 当注释被修改
- 当ASCII源文件第一次生成并编译
- 当块属性（“properties”对话框）被修改

接口时间标签

该时间标签会被刷新：

- 当接口改变（修改了数据类型或初始值、增加新参数）
- 如果接口结构改变，当ASCII源文件第一次生成并编译

该时间标记不会被刷新：

- 当符号被修改
- 当变量声明表中的注释被修改
- 当TEMP（临时）区域改变

块调用的规则

- 被调用的块的接口时间标签必须比调用块的代码时间标签旧。
- 只有当设有任何一个调用这个块的块被打开时，才可以修改这个块的接口。否则，修改了这个块之后再存储调用它的块，就无法从时间标签上识别这种不一致性。

时间标签冲突出现的过程

当调用块被打开时，时间标签冲突就显示出来。在对FC或FB接口作修改之后，在调用块中对这个块的所有调用以扩展的形式显示。

如果一个块的接口修改了，则所有调用该块的块都必须相应修改。

在对一个FB的接口修改后，已有的多重背景及背景数据块必须刷新。

15.4 共享数据块的时间标签

代码时间标签

该时间标签会被刷新：

- 当一个ASCII源文件首次生成时
- 当一个ASCII源文件编译时
- 当在块在declaration view或data view下作出修改时

接口时间标签

该时间标签会被刷新：

- 当在declaration view方式下，接口描述被修改（修改数据类型或初始值、有新参数）时。

15.5 背景数据块的时间标记

一个背景数据块用于存储功能块的形式参数和静态数据。

代码时间标签

背景数据块生成的时间和日期存储在这里。当在背景数据块的data view方式下输入实际值时，该时间标签被刷新。由于背景数据块的结构来自于相关的功能块（FB）或系统功能块（SFB），所以用户无法改变其结构。

接口时间标签

当一个背景数据块生成时，存储与其相关的FB或SFB接口时间标签。

无冲突打开的规则

FB/SFB及其相关的背景数据块的接口时间标签必须相匹配。

时间标签冲突出现的过程

如果修改了FB的接口，该FB的接口时间标签则被刷新。当打开一个与其相关的背景数据块时，则显示时间标签冲突的报告。因为此时背景数据块的时间标签与FB的不再匹配。在数据块的声明部分，接口显示由编译器生成的符号（伪符号）。该背景数据块此时只能看不能改。为改正这种类型的时间标签冲突，必须为修改过的FB重新生成背景数据块。

15.6 UDT 的以及源自于 UDT 的数据块的时间标签

用户定义的数据类型（UDT）可以用来生成大量的、具有相同结构的数据块。

代码时间标签

任何一种修改都会使代码时间标签刷新。

接口时间标记

当接口描述改变（修改数据类型或初始值、增加新参数），接口时间标签会被刷新。

当ASCII源文件编译时，UDT的接口时间标签也会被刷新。

无冲突打开的规则

- 用户定义的数据类型的接口时间标签必须比使用该数据类型的逻辑块的接口时间标签旧。
- 用户定义的数据类型的接口时间标签必须与源自于该UDT的数据块的时间标签一致。
- 用户定义的数据类型的接口时间标签必须比二级UDT的时间标签新。

时间标签冲突出现的过程

如果修改了一个UDT的定义，而该UDT被用于某个数据块、功能、功能块或另一个UDT的定义，当使用这个UDT的块被打开时，STEP 7将报告时间标签冲突。

UDT的元素被显示为一个展开结构。所有变量名被系统预置值覆盖。

15.7 更改功能、功能块或 UDT 中的接口

如果需要修改FC、FB或UDT中的接口，按下列步骤可避免时间标签冲突：

1. 从要修改的块以及所有的直接的或间接的块中生成一个STL源文件。
2. 保存所生成的源文件中的变化。
3. 编译所修改的源文件。

现在可以保存/下载接口的变化。

15.8 调用块时避免错误

STEP 7覆盖DB寄存器中的数据

当执行各种指令时，STEP 7将修改S7-300/400 CPU中的寄存器。当调用一个FB时，DB和DI寄存器中的内容将调换。这样可以在不丢时以前背景DB地址的情况下打开所调用的FB的背景DB。

如果采用绝对寻址方式，访问寄存器存储的数据时可能会发生错误。在一些情况下，寄存器AR1（地址寄存器1）中的地址和DB寄存器中的地址将被覆盖。这样可能会读/写错误的地址。



危险

当发生下面情况时可能会对财产和人身造成危险：

1. 使用CALL FC，CALL FB，CALL多重背景。
2. 用绝对地址访问DB(例如：DB20.DBW10)。
3. 访问复杂数据类型的变量。

可能会改变DB寄存器(DB和DI)、地址寄存器(AR1、AR2)以及累加器(ACCU1、ACCU2)中的内容。

此外，当调用FB或FC时不能使用状态字的RLO位作为附加参数使用。

保存正确数据

如果用缩写格式访问数据的绝对地址，DB寄存器中的内容会发生危险。例如，假设DB20打开(并且DB块号存储在DB寄存器中)，可以通过DBX0.2访问DB中的字节0的第2位数据，如果DB寄存器中的是不同的DB块号，则访问的数据不正确。

通过下列方法访问DB寄存器中的数据，可以避免发生错误：

- 用符号地址
- 用绝对地址(例如：DB20.DBX0.2)

如果用这种方法寻址，STEP 7将自动打开正确的DB。如果用AR1寄存器进行间接寻址，必须经常在AR1中调用正确的地址。

修改寄存器的条件

只有使用STL才能使用地址寄存器进行间接寻址。其它编程语言不支持间接寻址功能。

在所有的编程语言中，通过编译器修改DB寄存器也必须计算在内，以确保块调用时参数传递的正确性。

地址寄存器AR1和所调用块的DB寄存器的内容在下述情况可能被覆盖：

条　件	说　明
DB块作为实际参数	• 一旦使用DB块分配实际参数(例如DB20.DBX0.2)，STEP 7则打开相应的DB(DB20)并将DB块号写入DB寄存器的。在块调用后仍然生效。
在调用块时使用复杂数据类型	• 当在一个FC内调用一个块后，将向所调用的块传送一个复杂数据类型的形参元素(字符串、数组、结构或UDT)，AR1和所调用块的DB寄存器的内容被修改。 • 如果参数在VAR_IN_OUT区中，则在FB内调用的情况是相同的。
访问复杂数据类型的元素	• 当FB访问VAR_IN_OUT区中复杂数据类型的形参元素时(字符串、数组、结构或UDT)，STEP 7内部使用地址寄存器AR1和DB寄存器，它将修改两个寄存器中的内容。 • 当FC访问VAR_IN_OUT区中复杂数据类型的形参元素时(字符串、数组、结构或UDT)，STEP 7内部使用地址寄存器AR1和DB寄存器，它将修改两个寄存器中的内容。

注意：

- 当从版本1的块内调用一个FB时，如果调用前的命令没有限制RLO，则第一个布尔参数IN或IN_OUT的实际参数不能正确传送。在这种情况下，它是与现有的RLO组合的逻辑状态。
- 当调用一个FB(单个背景或多重背景)时，地址寄存器AR2被该写。
- 如果在一个FB中修改了地址寄存器AR2，则不能保证正确地执行FB。
- 如果没有将完全绝对寻址的DB地址传送给一个ANY参数，则ANY指针不会得到DB块号，而其得到的DB块号为0。

16 组态消息

16.1 消息的概念

消息（Message）功能允许在可编程控制器运行过程中快速地检测、定位及纠正错误，从而可显著地减少工厂的停工期。

要想有消息输出，首先要进行组态。

使用STEP 7，可以生成并编辑消息，这些消息与已被赋予消息文本和消息属性的事件相连。还可以编译这些消息并将它们显示在显示装置上。

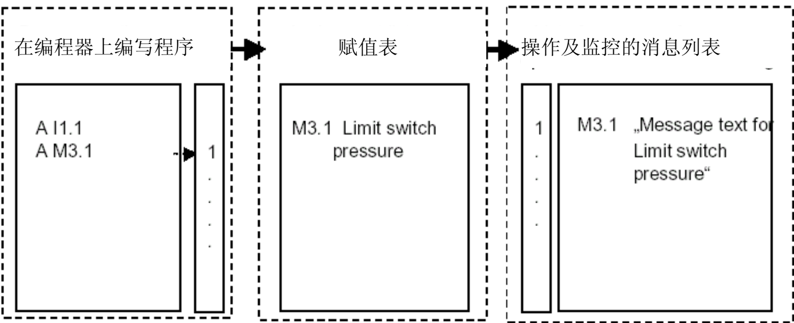
16.1.1 什么是不同的消息方法?

有生成消息的不同方法。

位消息

位消息需用编程人员完成以下三步：

- 在编程器上生成用户程序并设置所需要的位。
- 使用任意文本编辑器生成一张赋值表，在该表中一条消息文本被赋给一个消息位（如，M3.1=limit switch pressure）。
- 在赋值表的基础上，在操作面板上生成消息文本列表。

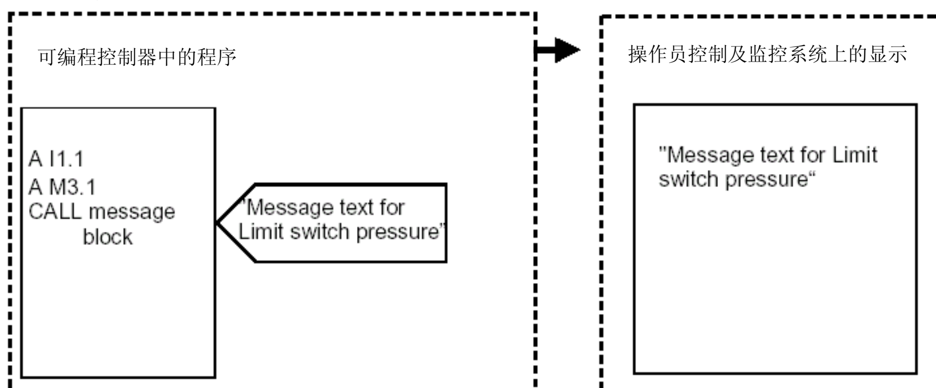


操作员接口系统不断地循环查询可编程控制器，查看是否有消息位发生变化。如果可编程控制器有信号变化，相应的消息则会显示。消息从操作员接口系统接收时间标签。

消息编号

消息编号功能只需要编程人员完成一个步骤：

- 在编程器上生成用户程序，设置所需要的位，并在编程时直接将所需的消息文本赋值给这个位。



可编程控制器不进行循环查询。当可编程控制器信号变化时，相应的消息号被传送到操作员接口系统，相应的消息文本被显示出来。该消息从可编程控制器得到接收时间标签，因此和位消息相比，它能够更准确地被追踪。

16.1.2 选择一种消息方法

概述

下表是不同消息方法的特性及要求：

消息编号	位消息
• 在编程器和操作员面板的公共数据库中进行消息处理 • 总线负载低（可编程控制器信号激活） • 消息从可编程控制器接收时间标签	• 编程器和操作员面板没有公共数据库 • 总线负载高（操作面板查询） • 消息从操作面板接收时间标签

消息编号方式可以识别以下三种类型的消息：

与块相关的消息	与符号相关的消息	用户定义的诊断消息
• 与程序同步 • 使用WinCC和ProTool显示（只有ALARM_S） • 可用于S7-300/400 • 用消息块编程： - ALARM，ALARM_8 - ALARM_8P - NOTIFY - NOTIFY_8P - ALARM_S(Q) - AR_SEND - ALARM_D(Q) • 传送到操作界面 - 通过AS-OS连接组态用于WinCC - 通过ProTool功能用于ProTool	• 与程序异步 • 使用WinCC显示 • 只用于S7-400 • 用符号表组态 • 通过系统数据块（SDB）传送到可编程控制器 • 通过PLC-OS连接组态传送至界面	• 与程序同步 • 在编程器的诊断缓冲区中显示 • 可用于S7-300/400 • 使用消息块（系统功能）编程 - WR_USMSG • 不传送到操作面板

STEP 7支持更加方便的消息编号方式，将在后续进行详细的描述。在这里介绍HMI设备中组态位消息。

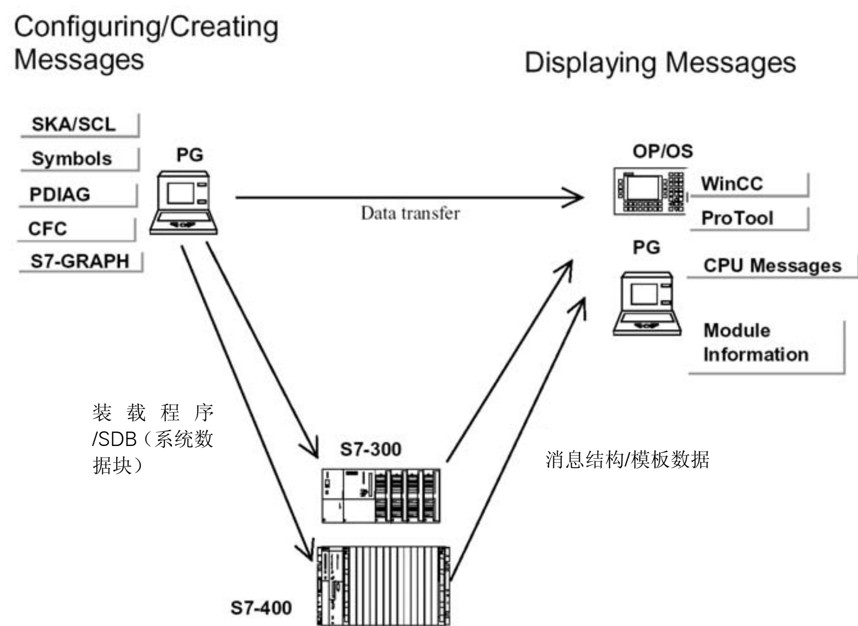
消息编码示例

消息方式	应用程序
与块相关的消息	用于报告与程序同步的事件，比如，显示控制器到达了一个定限值
与符号相关的消息	用于报告与程序无关的事件，比如，一个开关装置被监控
用户定义的消息	用于对每次SFC的调用，在诊断缓冲区中报告诊断事件

16.1.3 SIMATIC 组件

概述

下图所示为与组态和显示消息有关的SIMATIC组件的概观。



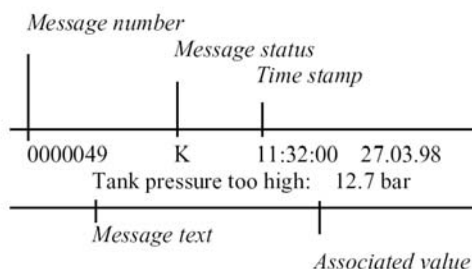
16.1.4 消息的组成

一条消息如何显示要看消息方式、使用的消息块以及显示设备。

部 件	描 述
时间标签	当消息事件出现时，在可编程控制器中生成
消息状态	可能有以下状态：到来，离开，离开无应答，离开有应答
相关值	有时消息可被赋予一个过程值，该值可被所用的消息块进行评估
图像	如果系统故障，出现的消息可随后显示在操作员站上
消息编号	由系统分配一个在整个项目或CPU中唯一的号码用以识别该消息
消息文本	用户组态

举例

下列所示，是操作面板上的一个报警消息。



16.1.5 可使用的消息块

可以选择下列消息块，每个消息块包含一个可编程的消息功能：

- SFB 33: “ALARM”
- SFB 34: “ALARM_8”
- SFB 35: “ALARM_8P”
- SFB 36 “NOTIFY”
- SFB 18: “ALARM_S” 和 SFC 17: “ALARM_SQ”
- SFB 37: “AR_SEND” (用于发送归档；不能组态消息文本和消息属性)
- SFB 31: “NOTIFY_8P”
- SFC 107: “ALARM_DQ”
- SFC 108: “ALARM_D”

详细信息请参见在线帮助。

什么时候使用哪个消息块

下表帮助您选择使用哪个消息块，选择的依据是：

- 块中可使用的通道号因此每个块调用有多少个信号被监控
- 消息是否需要确认
- 消息是否带有相关过程值
- 使用的显示设备
- CPU使用的项目数据

消息块	通道	确认	相关值	WinCC显示	ProTool显示	CPU 信息/S7 状态显示	PLC	解释
ALARM SFB33	1	可能	最多10个	有	无	无	S7-400	每个上升沿或下降沿到来时发送消息
ALARM_8 SFB34	8	可能	无	有	无	无	S7-400	一个或多个信号的每个到来或离开边沿发送消息
ALARM_8P SFB35	8	可能	最多10个	有	无	无	S7-400	同 ALARM_8
NOTIFY SFB36	1	无	最多10个	有	无	无	S7-400	同 ALARM
NOTIFY_8P SFB 31	8	无	最多10个	有	无	无	S7-400	As NOTIFY
AR_SEND SFB37	1			有	无	无	S7-400	用于发送文档不能组态消息文本和消息属性，
ALARM_SQ SFC17	1	可能	1	有	有*	有	S7-300 / S400	不是边沿变化而是每个SFC调用产生报文
ALARM_S SFC18	1	无	1	有	有*	有	S7-300 / 400	As ALARM_SQ
ALARM_DQ SFC 107	1	可能	1	有	有	有	S7-300 / 400	As ALARM_SQ
ALARM_D SFC 108	1	无	1	有	有	有	S7-300 / 400	As ALARM_SQ
*取决于OP类型								

16.1.6 形式参数、系统属性以及消息块

作为消息编号输入的形式参数

在用户程序中每个消息或每组消息需要一个形式参数，该形参在程序的变量声明表中定义为IN参数，从而可以使用这个形参作为消息编号输入，形成消息的基础。

如何配置形参的系统属性

作为开始组态消息的前提要求，必须首先按如下所述，用系统属性提供形式参数：

1. 为参数增加下述的系统属性：“S7_server”和“S7_a_type”
2. 根据在程序指令中调用的消息块，给系统属性赋值。赋给“S7_server”的总是“alarm_archiv”，而赋给“S7_a_type”的值要依据被调用的块而定。

系统属性和相应的消息块

消息块本身不会在消息管理器中作为一个对象显示出来；取而代之，显示中包含系统属性“S7_a_type”的值。这些值作为消息块具有相同的名字，以SFB或SFC的形式存在（“alarm_s” 例外）。

S7_a_type	消息块	描述	特　性
alarm_8	ALARM_8	SFB34	8通道，能被确认，无相关值
alarm_8p	ALARM_8P	SFB35	8通道，能被确认，每个通道最多10个相关值
notify	NOTIFY	SFB36	1通道，不能被确认，最多10个相关值
alarm	ALARM	SFB33	1通道，能被确认，最多10个相关值
alarm_s	ALARM_S	SFC18	1通道，不能被确认，最多1个相关值
alarm_s	ALARM_SQ	SFC17	1通道，能被确认，最多1个相关值
ar_send	AR_SEND	SFB37	用于发送文档
notify_8p	NOTIFY_8P	SFB31	8通道，不能被确认，最多10个相关值
alarm_s	ALARM_DQ	SFC107	1通道，不能被确认，最多1个相关值
alarm_s	ALARM_D	SFC108	1通道，能被确认，最多1个相关值

参考块的在线帮助，会得到更详细的信息。

如果在用户程序中所使用的消息块为具有相应系统属性的SFB或FB，并作为多重背景调用，系统属性将自动赋值。

16.1.7　消息模板及消息

消息组态允许按不同的步骤去生成消息模板或消息。至于采取何种步骤，要依据通过哪个消息模板块去访问消息组态而定。

消息模板块即可以是一个功能块（FB）也可以是一个背景数据块

- 使用FB可以生成一个消息模板作为生成消息的样板。为消息模板所作的所有条目都自动地输入到消息中。如果将一个背景数据块赋给一个功能块，则该背景数据块的消息按照消息模板和所分配的消息编号自动生成。
- 对背景数据块来说，可以修改消息，该消息是基于消息模板而生成的。

这里明显的差别是：消息编号是赋给消息的而不是给消息模板的。

为消息模板进行数据加锁

消息组态允许为事件驱动的消息输入文本和属性。比如，可以指定在特定的显示设备上怎样显示消息。为使消息生成更容易，可以使用消息模板。

- 当为消息模板输入数据（属性和文本）时，可以指定它们是否被加锁。若带有加锁属性则输入框旁边会增加一个键符。已加锁的文本在“Locked”栏中显示一个复选标志。
- 使用消息模板的“加锁数据”，不能在特定背景的消息中进行修改。数据只能被显示。
- 如果确实需要修改，必须回到消息模板，去掉锁，并在这里修改。这些修改不会自动加到以前生成的背景数据中。

修改消息模板的日期

当建立项目时，如果向项目或CPU所赋值的消息号为全局赋值，则消息模板的日期更改将影响背景数据。

- 对项目进行消息号赋值：当后续修改消息模板日期时，而且也想同样应用到背景数据时，必须修改背景数据的日期。
- 对CPU进行消息号赋值：后续修改消息模板日期将自动影响背景数据(例外：以前已修改了背景数据的日期或随后加锁或解锁消息模板日期)。

注意：

当向其他程序拷贝背景数据而没有拷贝消息模板时，背景数据只能部分显示，若要全部显示，必须将消息模板也拷贝到新程序中。

16.1.8 如何从消息模板块中生成STL源文件

当从消息模板块中生成STL源文件时，组态信息同时也写入该源文件。

该信息写入pseudo-comment，用“$ALARM_SERVER”开始，用“*”结束。

注意

当为块设置符号参考时，注意在编辑源文件前不能修改符号表。

当源文件含有多个块时，几个pseudo-comment将和并为一个注释块。带有消息属性的单独块不能从STL源文件中删除。

16.1.9 分配消息号

当为项目或CPU分配消息号时，可以指定这些消息号。对CPU分配消息号可以在不改变消息号的情况下拷贝程序，在这种情况下需要对其重新编译。对于CPU，如果使用WinCC V6.0或ProTool V6.0软件在HMI设备上只可能显示消息号。如果使用先前的HMI软件，必须为项目选择消息号。

16.1.10 对面向项目和面向CPU分配消息号的区别

下表列出了对项目和CPU进行消息号赋值的区别：

项　目	CPU
一些消息属性和文本取决于所使用的HMI设备，必须设置其显示特性	属性及文本的赋值不用根据所使用的HMI设备而决定
复制后必须重新编译程序	可将程序拷贝到一个项目的其它地方或拷贝到其它项目中。如果只拷贝了单个块，则程序必须重新编译
当以后修改消息类型数据时(文本和属性)，必须修改背景	当以后修改消息类型数据时(文本和属性)，将自动修改背景(例外：如果在以前改变了背景)
文本只能写入一行	文本可以写入多行

16.1.11 修改一个项目消息号赋值的选项

在SIMATIC管理器的“Message number”标签中可以为以后的项目以及库预置消息号(**Options > Customize**)。在该标签中可以决定是否消息号只分配给CPU或只分配给项目。如果想在以后赋值，可以选择“Always ask for setting”。

如果在创建项目或创建库时初始的却省设置为“CPU-oriented”或“ project- oriented”处于激活状态，则不再需要改变为该项目或库修改消息号赋值类型。

如果原来设置为“project- oriented ”，现在想设置为“CPU-oriented ”，则按下列步骤进行：

1. SIMATIC管理器中选择相应的项目或相应的库。
2. 选择菜单命令**File > Save As**。
3. 在下一个对话框中选择“With rearrangement”，并输入一个新名字。
4. 点击“OK”。
5. 在下一个对话框中可以指定“CPU-oriented”唯一消息号分配。

可以用**File > Delete**命令删除原始项目或库。“CPU-oriented”和“ project- oriented”消息号的分配是不同的。

16.2 组态面向项目的消息

16.2.1 如何分配面向项目的消息号

在整个项目中，每个消息有一个唯一的号码，通过该号码进行识别。为了对其归档，每个

STEP 7程序被分配一定范围的号码，该号码在总范围(1至2097151)之内。如果要分配的号码已被占用，则新程序必须分配一个新的号码段。如果发生号码段冲突，STEP 7将自动打开可以指定新号码段的对话框。

如果还没有组态消息，也可以用Edit > Special Object Properties > Message Numbers为一个S7程序设置或改变号码段。

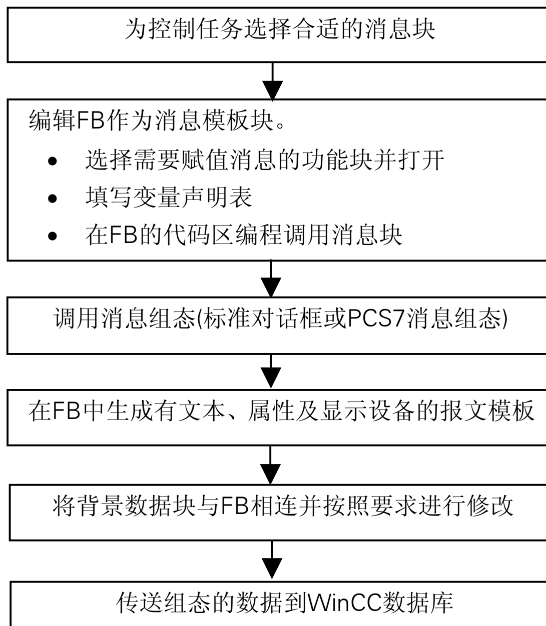
缺省时，消息的号码段以20000为步进单位。

16.2.2 赋值和编辑与块相关的消息

与块相关的消息被赋值到一个块的背景数据块。可以使用系统功能块SFB和系统功能SFC作为消息块，来创建一个与块相关的消息。

16.2.2.1 如何生成与块相关的消息（面向项目）

基本步骤



编程消息模板块（FB）

1. 在SIMATIC管理器中选择一个需要生成与块相关消息的功能块（FB），双击打开它。

   结果：打开被选中的块并显示在“LAD/STL/FBD”窗口。
2. 填写变量声明表。因为对每一个被调用的消息块，必须在调用功能块中声明变量。

在变量声明表中，在“Declaration”栏中输入下列变量：

- 在参数“IN”中输入一个符号名作为消息块的输入，例如，“Meld01”（为消息输入01）以及数据类型（必须是“DWORD”，没有初始值）。
- 在参数“STAT”中为调用的消息块输入符号名，例如，“alarm”及相应的数据类型，这里为“SFB33”。

3. 在功能块的指令部分，插入对选中的消息块的调用，这里为“CALL alarm”，用RETURN键完成输入。

   结果：被调用的消息块（这里是SFB33）的输入变量显示在功能块的代码区。

4. 将第2步中为消息块输入分配的符号名，这里是“Mess01”赋给变量“EV_ID”并确认系统属性被用于消息组态。

   结果：如果“Name”栏没有选中，则在该栏中会出现一个标志。选中的块则被设置为消息模板块。所要求的系统属性（如S7_server和S7_a_type）及相应值会自动被赋值。(注意：对于当前的SFC，只能自己为“IN”参数分配系统属性。方法是通过菜单命令**Edit > Object Properties**并选择“Attributes”)。

   注意：如果没有调用SFB，而是调用了含有多重背景和组态消息的FB，则必须在调用的块里用多重背景对该FB的消息进行组态。

5. 重复步骤2至4，在功能块中调用所有的消息块。
6. 用菜单命令**File>Save**保存该块。
7. 关闭“LAD/STL/FBD”窗口。

打开消息组态对话框

- 在SIMATIC管理器中选择菜单命令**Edit > Special Object Properties > Message**。

结果：STEP 7消息组态对话框（标准对话框）打开。在PCS 7消息组态下可以找到有关打开PCS 7消息组态功能的信息。

生成消息模板

1. 选择需要的消息块并且在“Attributes”和“Text”选项栏中输入所需的属性和消息文本，或选择消息属性。

   如果选择的是一个多通道消息块（例如，“ALARM_8”），可以分配指定的文本，对于确定的扩展，定义每个子编码的属性。

2. 击“New Device”按钮将所需的显示设备赋给消息模板，并在“Add Display Device”对话框中选择所需的显示设备。

在下面的制表页中为显示设备输入所需文本及属性。用“OK”退出对话框。

注意

当编辑特定的显示设备文本和属性时，请阅读随随显示设备提供的文件。

生成背景数据块

1. 当生成消息模板时，可以给它关联一个背景数据块，并且为这些数据块编辑特定背景消息。要实现这一步，需在SIMATIC管理器中打开一个块，在该块中要调用前面组态的功能块，例如，双击“OB1”。在该OB块的指令部分输入调用指令（“CALL”），FB的名字和号码以及作为背景与FB相关的背景数据块。使用RETURN键确认。

   例如：输入“CALL FB1，DB1”。如果DB1不存在，对是否生成背景数据块的提示回答“Yes”。

   结果：背景数据块生成了。在OB的指令部分显示相关FB的输入变量，这里的示例是“Mess01”，以及由系统分配的消息编号，这里为“1”。

2. 用菜单命令**File>Save**，存储OB，并关闭“LAD/STL/FBD”窗口。

编辑消息

1. 在SIMATIC管理器中，选择已生成的背景数据块，例如“DB1”，选择菜单命令**Edit>Special Object Properties>Message**，打开消息组态对话框。

   结果：“Message Configuration”对话框打开，显示所选中的带有由系统分配的消息编号的背景数据块。

2. 在适当的标签栏中对相应背景数据块作必要的修改，如果需要，可增加其它的显示设备。点击“OK”退出对话框。

   结果：为选中的背景数据块所作的消息组态就完成了。

传送组态数据

- 将组态的数据传送至WinCC数据库（通过AS-OS连接组态）或ProTool数据库。

16.2.2.2 如何编辑与块相关的消息

1. 在SIMATIC管理器中，先选择一个块，然后选择菜单命令**Edit > Special Object Properties > Message**。
2. 在文件夹结构中，点击一个消息块输入或消息子号中的一个(如果有的话)。

   结果：消息的标签部分被显示。

3. 分别在“Text”和“Attributes”栏内输入所需的文本和属性。

   结果：在所有的显示设备上显示已经创建的标准消息。

4. 使用“New Device”按钮，加入一个“ProTool”(Opx)或“WinCC”类型的新的显示设备。所组态的消息只能在这些显示设备上显示。

   结果：新设备被选择和添加，对应的消息标签被显示。

5. 在“Texts”和“Attributes”栏中输入显示指定的消息的文本和属性。

   结果：已生成的消息变量只能用在所选择的显示设备上。

如果要在现有的显示设备上编辑其它消息：

- 在详细视窗中通过双击打开消息块。

  结果：自动选择第一个显示设备，可以针对该设备编辑与显示相关的消息。

16.2.2.3 如何对PCS 7消息进行组态（面向项目）

为编辑消息模板以及将消息发送到WinCC显示设备，STEP 7中的PCS7消息组态功能提供用户友好方式如下：

- 简化了显示设备组态（自动生成）
- 简化了消息属性和文本输入
- 确保消息标准化

打开PCS 7消息组态功能

1. 在SIMATIC管理器中，选择需要编辑消息文本的块（FB或DB），使用菜单命令**Edit > Object Properties**，打开对话框进入系统属性。
2. 在显示的表格中输入系统属性“S7_alarm_ui”以及值“1”（数值“0”将关闭PCS7消息组态）。在LAD/STL/FBDP中可以设定属性参数。以后生成并赋值到相应FB的DB，将采用这些属性，并且可以切换，与其消息类型无关（FB）。

注意

当你输入系统属性时，语法检查正在运行，不正确的输入被标定为红色。

---

3. 用“OK”退出对话框。
4. 选择菜单命令**Edit>Special Object Properties>Message**。

   结果：“PCS7 Message Configuration”对话框被打开。

编辑消息模板

1. 在SIMATIC管理器中，选择需要编辑消息文本的FB，打开PCS7消息组态对话框。

   结果：对话框为每一个在FB中声明了变量的消息块显示一个标签。
2. 为消息的组件“Origin”、“OS area”及“Batch ID”填写文本框。
3. 输入消息级别并为消息块使用的所有事件输入事件文本，指定每个事件是否必须一一确认。
4. 对于用于所有背景的、不能被修改的消息组件，点击“Locked”复选框。

编辑消息

1. 在SIMATIC管理器中，选择需要编辑消息文本的背景数据块，打开PCS7消息组态对话框。
2. 不要修改未加锁的由背景定义的消息部分。

16.2.3 设定和编辑与符号相关的消息

16.2.3.1 如何赋值和编辑与符号相关的消息

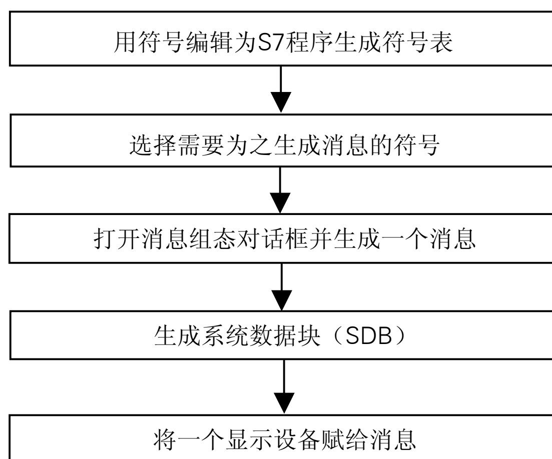
在符号表中符号相关的消息（SCAN）被直接赋值给信号。所允许的信号都是布尔地址：输入（I）、输出（Q）及位存储（M）。可以用消息组态功能给这些信号赋予不同的属性、消息文本及多达10个的相关值。可以通过设置过滤功能在符号表中轻松地选择信号。

用符号相关消息，可以以一个预先设定的时间间隔来扫描信号，从而判断信号是否有变化。

注意

时间间隔与所用的CPU有关。

基本步骤



在处理过程中，对已组态消息的信号进行扫描与程序执行是异步的。这种扫描以组态的时间间隔进行，消息显示在指定的显示设备上。

注意：

如果要对于符号相关的消息进行赋值或编辑，与此同时，已经在两个符号表中对符号进行了拷贝，则此时必须首先关掉不再使用的那个符号表。否则将不能保存消息组态。在特定条件下，可能会丢失消息组态对话框中的最后一条输入。

16.2.4 生成并编辑用户定义的诊断消息

使用这一功能，可以在诊断缓冲区中写入用户条目，并发送在消息组态程序中生成相应的

消息。用户定义的诊断消息可通过系统功能SFC52（WR_USMSG：错误等级A或B）生成。必须在用户程序中插入对SFC52的调用并给它分配事件ID。

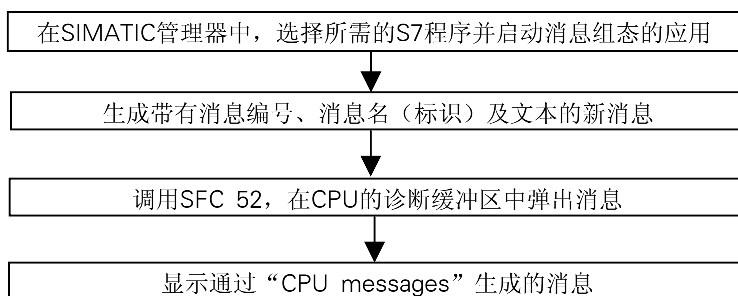
要求

在生成用户定义的诊断消息之前，必完成以下事项：

- 在SIMATIC管理器中生成项目
- 在项目中生成S7/M7程序，消息将被分配给该程序

基本步骤

生成并显示用户定义的诊断消息，可按以下步骤进行：



16.3 组态面向 CPU 的消息

16.3.1 如何分配面向CPU的消息号

CPU的消息号是唯一的。对每个CPU分配一个消息号区段。与对项目分配消息号相比，对新程序不需要分配一个新的消息号区段。因此不需要对程序进行新的编译。但当拷贝单个块时，则必须重新编译程序。

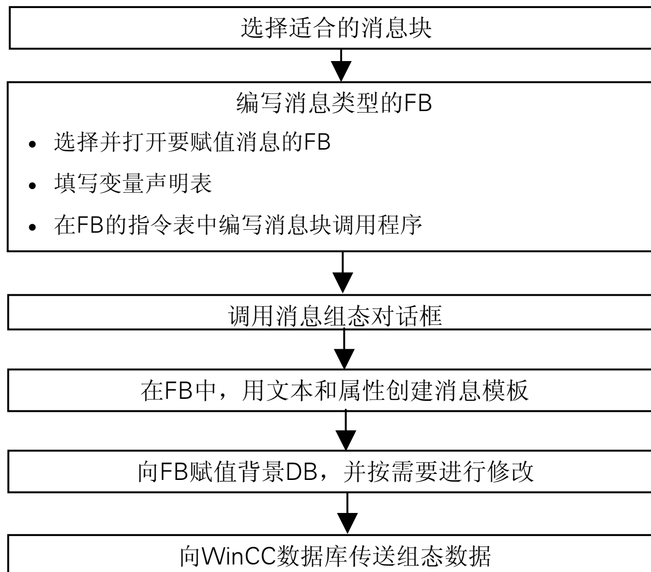
要求

- WinCC V6.0
- ProTool V6.0

16.3.2 赋置和编辑与块相关的消息

16.3.2.1 如何为CPU创建与块相关的消息（面向CPU）

操作原理



编写消息模板块(FB)

1. 在SIMATIC管理器中，选择需要生成与块相关消息的功能块FB，并双击打开它。

   结果：所选的功能块被打开，并显示在“LAD/STL/FBD”窗口中。

2. 填写变量声明表。必须为每个在功能块中进行调用的消息声明相应的变量。

   在变量概述栏中输入下列变量：

   - 对于参数“IN”，输入一个符号名作为消息块输入，例如“Meld01”(消息输入01)以及数据类型(必须是没有初始值的“DWORD”)。
   - 对于参数“STAT”，输入一个符号名作为消息块调用，例如“alarm”以及相应的数据类型，在此为“SFB33”。

3. 在功能块的程序中，插入调用所选择的消息块，在此为“CALL alarm”，并按回车键。

   结果：所调用消息块的输入变量显示在功能块的程序中。

4. 赋值在第2步中已分配的符号名。对于消息块输入，此处为“Mess01”赋值给变量“EV_ID”。

   结果：如果没有选择“Name”栏，则在该栏中出现一个标志。所选择的块被设置为消息类型块。所需的系统属性(例如S7_server和S7_a_type)以及相应的值将自动赋值。(注意：对于一些特定的SFC，只能自己对“IN”参数进行系统属性赋值，通过选择**Edit > Object Properties**并选择“Attributes”选择框实现)。

注意：如果所调用的FB含有多个背景，并且已经对所代替的SFB组态了消息，也必须在调用的块中为该FB组态消息。

5. 重复第2至4步，实现功能块中所有的消息块调用。
6. 用**File > Save**进行保存。
7. 关闭“LAD/STL/FBD”窗口。

打开消息组态对话框

- 选择所需的消息块，并使用SIMATIC管理器中的**Edit > Special Object Properties > Message**。

  结果：STEP 7消息组态对话框被打开。打开PCS 7消息组态功能可以在PCS7消息组态中得到。

编辑消息模板

- 选择所需的消息框。
- 输入所需的文本或属性

  在“Message Configuration”对话框中，点击“More”按钮可以在“Default Texts”表中输入消息文本和其它文本。如果选择了多通道消息块(例如“ALARM_8”)，可以为每个子号输入特殊文本及属性。
- 如果不想改变背景的文本或属性，可以在消息模板中对其加锁。

创建背景数据块

1. 建立完消息模板后，可以对消息模板创建背景数据块并对这些数据块的背景消息进行编辑。在SIMATIC管理器中打开程序块，该程序块调用了以前组态好的功能块，例如双击打开“OB1”。在打开的OB1程序中输入调用指令(“CALL”)，写好调用FB的号码及名称，以及FB所使用的背景数据块，按回车确认。

   举例：输入“CALL FB1，DB1”。如果DB1不存在，会出现是否要创建背景数据块的提示符，点击“Yes”。

   结果：创建背景DB。在OB的程序中将显示FB的输入变量(例如“Mess01”)以及系统所分配的消息号(例如：“1”)。
2. 用**File > Save**保存OB并关闭“LAD/STL/FBD”窗口。

编辑消息

1. 在SIMATIC管理器中选择创建好的背景DB(例如：“DB1”)，用菜单命令**Edit > Special Object Properties > Message**打开消息组态对话框。

   结果：“Message Configuration”对话框被打开，显示系统所分配的背景DB的消息号。
2. 可以对相应的背景数据块进行必要的修改并增加显示设备。点击“OK”退出对话框。

   结果：所选择的背景数据块的消息组态已完成。

注意：

如果消息背景的文本和属性显示绿色，则表示在消息模板中文本与属性正在配置，在消息背景中不能修改。

传送组态数据

- 将组态数据传送到WinCC数据库(通过AS-OS连接)或ProTool数据库。

16.3.2.2 如何编辑与块相关的消息（面向CPU）

1. 选择消息块，然后选择菜单命令**Edit > Special Object Properties > Message**来调用消息组态。
2. 在“Default Texts”和“Additional Texts”栏中输入所需的文本。也可以点击“More”按钮并在“Default Texts”和“Additional Texts”对话框中输入所需的文本。

   结果：建立了一个标准消息。

注意：

如果消息背景的文本和属性显示绿色，则表示在消息模板中文本与属性正在配置，在消息背景中不能修改。

16.3.2.3 如何组态PCS 7消息（面向CPU）

对于编辑消息模板以及在WinCC(V6.0)上显示要输出的消息，STEP 7中的PCS 7 消息组态功能提供用户友好的方法：

- 简化了显示设备的组态
- 简化了消息的输入属性和输入文本
- 保证消息的标准化

打开PCS 7消息组态功能

1. 在SIMATIC管理器中选择要编辑的FB块或DB块。通过菜单命令打开**Edit > Object Properties**打开系统属性对话框。
2. 在显示的表格中，输入系统属性“S7_alarm_ui”并输入值“1”(0为禁止PCS7消息组态工具)。可以用LAD/STL/FBD设置属性参数。随后生成的DB以及分配的FB将使用这些设置，也可以被切换使用本身的属性，独立于消息模板。

注意：

当输入系统属性时会进行语法检查。错误的输入会以红色高亮显示。

3. 点击“OK”退出对话框。
4. 选择菜单命令 **Edit > Special Object Properties > Message**。

   结果：打开“PCS 7消息组态”对话框。

编辑消息模板

1. 在SIMATIC管理器中，选择要编辑消息文本的FB，并打开PCS 7消息组态对话框。
2. 点击“More”按钮打开“Message text block”。在“Origin”、“OS area”和“Batch ID”中输入文本。
3. 为所有消息块的事件输入消息等级和事件文本，并指定是否对每个事件进行单独确认。
4. 对适用于所有背景并且不会改变的消息部分，选择“Locked”选择框。

编辑消息

1. 打开SIMATIC管理器，选择对其消息进行编辑的背景数据块，并打开PCS7消息组态功能。
2. 不要编辑没有加锁背景定义的消息部分。

16.3.3 赋值和编辑与符号相关的消息

16.3.3.1 如何赋值和编辑与符号相关的消息（面向CPU）

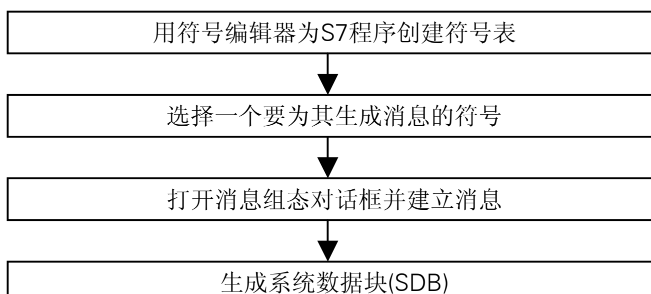
与符号相关的消息可直接赋值到符号表中的信号。所允许的信号全部是布尔地址：输入(I)、输出(Q)和位存储器(M)。可以对这些信号分配不同的属性和消息文本，以及最多10个与消息相关的过程值。通过设置过滤器可以很方便地在符号表中选择这些信号。

对于一个与符号相关的消息，可以在一个预置的时间间隔内对信号进行扫描，以决定信号是否发生了变化。

注意：

时间间隔与使用的CPU有关。

基本步骤



在处理期间，扫描已组态消息的信号与程序执行是异步的。在组态的时间间隔进行扫描。消息将在所分配的显示设备上显示。

注意：

如果需要对于符号相关的消息进行赋值或编辑，与此同时，已经在两个符号表中对符号进行了拷贝，则此时必须先关掉不再使用的那个符号表，否则将不能保存消息组态。在特定条件下，可能会丢失消息组态对话框中的最后一条输入。

16.3.4 赋值和编辑用户定义的诊断消息

使用这一功能，可以在诊断缓冲区，并发送在消息组态程序中生成相应的消息。用户定义的诊断消息可通过系统功能SFC52（WR_USMSG：错误等级A或B）生成。必须在用户程序中插入对SFC52的调用并给它分配事件ID。

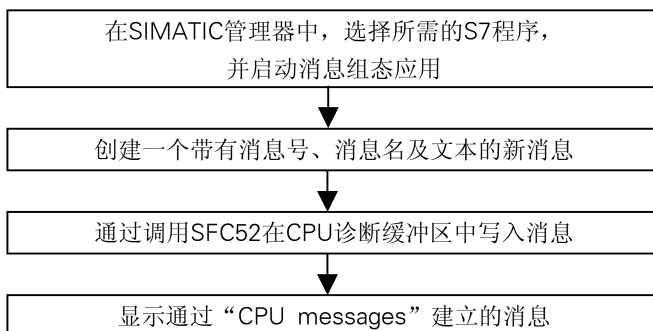
要求

在建立用户定义的诊断消息前，必须完成下列任务：

- 在SIMATIC管理器中建立好一个项目
- 在项目中建立好一个S7/M7程序，对该程序赋值一个或多个消息

基本步骤

建立和显示用户定义的诊断消息的步骤如下：



16.4 编辑消息的技巧

16.4.1 向消息加入关联值

为了将当前信息(例如来自过程的信息)加入到与块相关的和与符号相关的消息中，可以将关联值加入到消息文本中的任何位置。

加入关联值的步骤如下：

1. 建立一个新块，结构如下：
   @<关联值号>[<单元类型>]<格式代码>@
2. 将该块插入到消息文本中要显示该关联值的地方。

单元类型

元素类型	数据类型
Y	字节
W	字
X	双字
I	整数
D	整数
B	布尔类型
C	字符
R	浮点

元素类型唯一指定了通过PLC传送的数据类型，不能用于计算操作。

格式代码

这些代码指定了显示设备上的关联值的输出格式。格式指令以“%”赋号为前缀。对于消息文本，有以下固定的消息代码：

格式代码	说明
%[i]X	带 i 索引符的十六进制值
%[i]u	带 i 索引符的无符号的十进制值
%[i]d	带 i 索引符的有符号的十进制值
%[i]b	带 i 索引符的二进制值
%[i][.y]f	整数(定点数) 带[-]dddd.dddd格式的符号值 dddd：
%[i]s	字符串(ANSI串)
%t#<文本库的名称>	访问文本库

如果格式代码太小，则值将以全部长度输出。如果格式代码太大，则输出的值前面会出现适当空隔。

相关值举例

@1%6d@：相关值1的值以十进制显示，最多6位。

@2R%6f@：相关值2的值，例如“5.4”以整数形式显示“5.4”(前三位为空)。

@2R%2f@：相关值2的值，例如“5.4”以整数形式显示“5.4”(数位太少，不会发生截断情况)。

@1W%t#Textbib1@：数据类型为WORD的相关值1在文本库TextLib1中作为文本的索引。

注意：

当使用S7-PDIAG时，单元类型必须始终为“X”。如果要将多个相关值赋给一个ALARM_S块，则可以发送一个最长为12字节的数组。例如一个最长12字节的字符，一个最长6个字或整数数据，一个最长3个双字、浮点或双整形数数据。

16.4.2 将文本库中的文本集成到消息中

可以将最多4个不同文本库中的文本集成到一个消息中。文本可以自由放置，也可用外语消息。

步骤如下：

1. 在SIMATIC管理器中，选择CPU并选择菜单命令Options > Text Libraries > System Text Libraries或Options > Text Libraries > User-Specific Text Libraries打开文本库。

注意

如果已选择了将消息号赋值给CPU，则只能将用户文本库中的文本集成到消息中。（面向CPU的消息号）

---

1. 确定要集成的文本。
2. 在消息中需要显示文本的地方输入 @[Index]%t#[Textbib]@格式的占位符。

注意：

[Index]=1W，1W是WORD类型消息的第一个相关值。

举例

所组态的文本：Pressure rose @2W%t#Textbib1@

文本库的名称Textbib1：

索引	德语	英语
1734	Zu hoch	Too high

第二个相关值赋值为1734。显示的消息为：Pressure rose too high。

16.4.3 删除相关值

可以在消息文本中删除表示相关值的字符串来删除相关值。

步骤：

1. 选择要删除相关值的消息文本中的信息块。该块以@符号开始，并以@符号结束。
2. 从消息文本中删除该信息。

16.5 翻译并编辑与操作员相关的文本

在过程编辑当中输出到显示设备上的文本通常用与自动化任务编程相同的语言输入。

也许会有这样的情况，对显示设备上的消息进行反应的操作员却不讲这种语言。这样的用户需要的是他自己语言的文本，以确保顺利、无障碍地处理并对系统输出的消息快速反应。

STEP 7允许将任意的、所有操作员相关的文本翻译成所需的语言。唯一的前提条件是已在项目中安装了该语言（SIMATIC管理器中的菜单命令Options> Language for Display Devices）。可用的语言的数量由安装Windows 时决定（系统特性）。

用这种方法可以确信，以后任何用户面对这条消息都可以让它以适当的语言显示。这一系统特征明显地增加了过程的安全性和准确性。

16.5.1 翻译并编辑用户文本

对整个项目、S7程序、块文件夹、单个块或符号表来说，如果在这些对象中有组态的消息，则可生成用户文本列表。例如它们包含所有文本和消息，可以在设备上显示。对于一个项目，可以有一些操作相关文本列表，可以将它们翻译成所需的语言。

可以在一个项目中选择已有的语言（菜单命令：Options > Language for Display Devices...）。还可以在之后增加或删除语言。

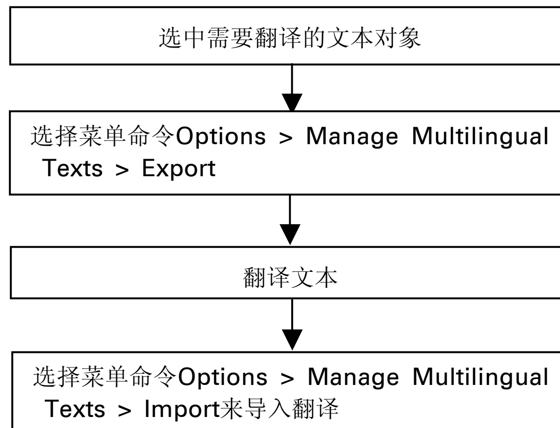
导入和导出与操作员相关的文本

可以翻译或编辑相关的操作文本，无论它是在STEP 7内生成的还是在STEP 7之外生成的。要实现这一点，将所显示的相关的操作文本列表导出到一个可编辑的文本文件中，这个文本文件可以在ASCII编辑器或象Microsoft Excel这样的表格编辑器中编辑（菜单命令Options>Manage Multilingual Texts>Export）。打开该文件后，屏幕上将显示包含各种语言的文表，其中第一栏显示所设定的标准语言。当文本翻译完后，将文本导回STEP 7。

在项目哪个部分导出的相关操作文本，只能从这个部分导入。

基本步骤

在SIMATIC管理器中，使用菜单命令Options > Language for Display Devices设置需要转换语言的用户文本。



注意：

只能使用翻译所使用的用户程序打印用户文本。

16.6 翻译和编辑文本库

16.6.1 用户文本库

一个用户文本库可以根据相关值动态查看文本或文本段。在此，相关值为当前文本提供文本库索引。在要显示动态文本的地方输入一个占位符。

可以为程序创建用户库，在其中可以输入文本和选择自己的索引。应用程序将自动检查用户库中索引的唯一性。该CPU中可以使用的所有消息均与用户文本库中有交叉参考。

文本库文件夹中的文本库的数量是没有有限的。因此可以用同一个程序控制不同的控制任务或仅仅改变文本库符合应用的需求。

注意：

当将含有交叉参考的消息模板块复制到另一个程序的文本库中时，必须要包含相应的文本库，或者建立一个相同名字的新的文本库或在消息文本中编辑交叉参考。

当创建一个文本条目，在缺省的条件下分配一个索引。当进入一个新的文本行，下一个没有使用的索引号作为缺省值分配。文本库索引必须是明确的，否则被应用拒绝。

16.6.2 创建用户文本库

创建用户文本库，步骤如下：

1. 在SIMATIC Manager中选择Program或Program中需要创建用户文本的下级对象，选择菜单命令Insert > Text Libraries > Text Library Folder。
   结果：创建“Text Library”文件夹。
2. 选择“Text Library”文件夹，选择菜单命令Library >User Text Library并命名文本库。
3. 打开新的文本库，选择菜单命令Options >User Text Library。
4. 现在可以输入需要的文本。

注意：

当创建一个文本条目，在缺省的条件下分配一个索引。当进入一个新的文本行，下一个没有使用的索引号作为缺省值分配。文本库索引必须是明确的，否则被应用拒绝。

16.6.3 怎样编写用户文本库

编辑存在的用户文本库，步骤如下：

a. 在SIMATIC Manager中选择Program或Program中需要编辑用户文本的下级对象，选择菜单命令Options > Text Libraries>User Text Library。
b. 选择需要从“Available Text Libraries ”对话框打开的文本库。
c. 编辑显示的文本，可以使用不同的编辑功能（例如，Find和Replace）。可以输入需要的文本，必须实时修改为文本自动产生的索引号，如果偶然输入以前分配的索引号，索引号将变为突出的红色。插入新的一行，选择手动命令Insert >New Row或点击工具栏相应的图标。
d. 如果需要Hardcopy，打印需要的文本。
e. 在完成所有的任务后关闭用户文本库。
f. 在完成所有文本编辑后，关闭编辑的应用。

注意：

当将含有交叉参考的消息模板块复制到另一个程序的文本库中时，必须要包含相应的文本库，或者建立一个相同名字的新的文本库或在消息文本中编辑交叉参考。

16.6.4 系统文本库

当生成一个新块时将自动建立系统文本库，例如在“Report System Errors”中。用户不能自己创建系统文本库，只能编辑现有的文本库。

CPU中所有可用的消息对于文本库带有交叉参考功能。

16.6.5 翻译文本库

系统文本库和用户文本库提供的文本列表可以集成在消息中，在运行时动态刷新，显示在编程器或其它显示装置上。

系统文本库中的文本可以由STEP 7或STEP 7可选软件包提供。可以将几个文本库赋值给一个CPU。可将这些文本翻译成所需语言。

在SIMATIC Manager中，可以选择适用于项目的语言（菜单命令**Options > Language for Display Devices...**），也可以添加或删除语言。

当开始翻译文本库时(菜单命令**Options > Manage Multilingual Texts > Export**)，将生成一个导出文件，可以用EXCEL对其进行编辑。当打开该文件后，在显示屏上将显示一个表。表的每一栏都表示一种语言。

注意：

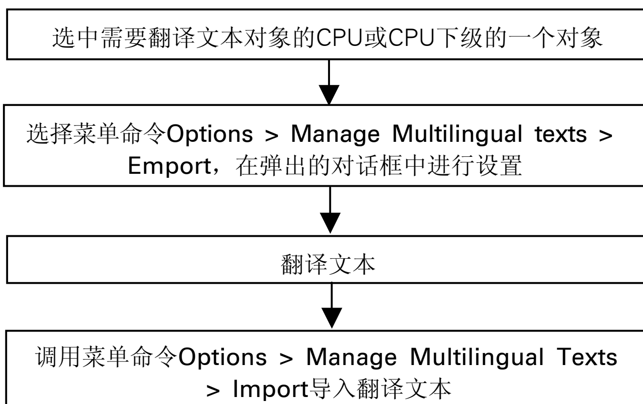
不能用双击该文件的方法打开该*.cvs输出文件，只能用EXCEL的**File > Open**命令打开。

导出文件举例

德文	英文
ausgefallen	Failure
gostort	Disruption
Parametrierfehler	Faulty parameter assignment

基本步骤

在SIMATIC Manager中，确信使用菜单命令 **Options > Language for Display Devices** 设置需要翻译的用户文本所需的语言。



16.7 传送消息组态数据到可编程控制器

概述

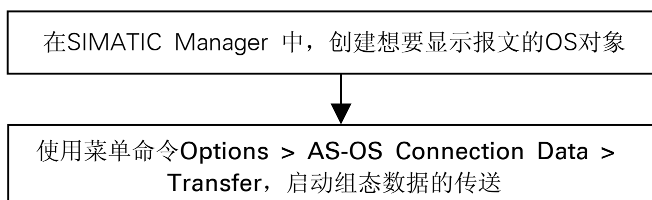
使用传送程序AS-OS Engineering，可将生成的消息组态数据送至WinCC数据库。

要求

在开始传送之前，下列要求必须满足：

- 已安装了“AS-OS Engineering”程序。
- 已为创建的消息生成组态数据。

基本步骤



16.8 显示 CPU 消息和用户定义的诊断消息

用“CPU Message（CPU消息）”功能(菜单命令**PLC > CPU Messages**)，可以显示诊断事件的异步消息和用户定义的诊断消息以及ALARM_S块的消息(SFC18和SFC108生成与块相关的消息总是需要确认，SFC17和SFC107生成与块相关的消息可以被确认)。

还可以通过使用菜单命令 **Edit > Message > User-Defined Diagnostics** 从CPU Messages应用程序中启动消息组态应用程序，并且生成用户定义的诊断消息。这就需要通过一个在线项目启动CPU Messages应用程序。

显示选项

用“CPU Messages”功能，可以决定所选CPU的在线消息是否显示以及如何显示：

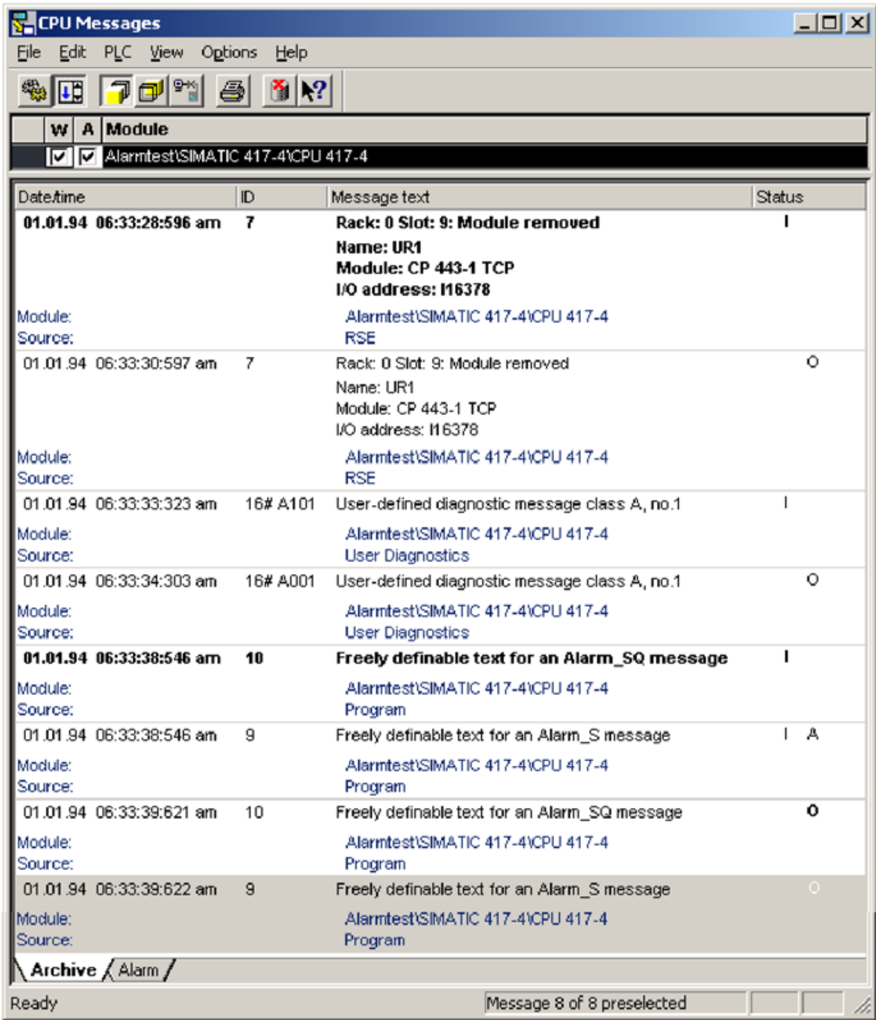
- “Highlight in the Task Bar”：一接收到消息，消息窗口不出现在前台，CPU消息在视窗的任务栏中高亮显示。
- “**Leave in the Background**”：CPU消息被接收到后台中。当收到新消息时该窗口仍处于后台中，如果需要可以被提到前台。
- “**Ignore Message**”：CPU消息不显示，并且与前两种模式相比，不存档。

在“CPU Messages”窗口，可以选择“Archive”或“Interrupt”。两种选择中，可以选择菜单命令**View > Display Info Text**来指定消息显示时是否带有信息文本。用户可以按需求分类消息列。

“Archive”标签

进来的消息被显示和归档，并根据时间进行排序。通过菜单命令Options > Settings，在“Settings-CPU Messages”对话框中设置归档的容量(40至3000条CPU消息)。如果超过设定的容量，则删除最早的消息。

可确认的消息(ALARM_SQ和ALARM_DQ)以加粗的形式显示。可以在菜单命令Edit > Acknowledge CPU Message下对这些消息进行确认。



“Interrupt”标签

在“Interrupt”中显示还没有收到的或还没有确认的发自ALARM_S块消息的状态。

通过菜单命令View > Multiline Messages在一行或多行上显示消息。此外，可根据需要对其排序。

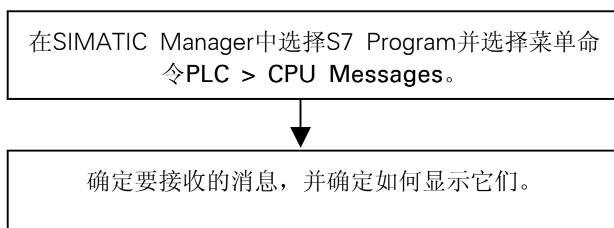
更新ALARM_S块的消息

在更新期间，所有未发送的或未确认的消息重新归档。以下情况更新消息：

- 与消息相关的模板执行重启动(不是冷启动)。
- 在模板列表中点击ALARM_S块消息的“A”选项。

基本步骤

为所选的模板组态CPU消息。



16.8.1 组态CPU消息

要为所选择的模板组态CPU消息，可如下进行：

1. 在SIMATIC Manager中，通过一个在线项目起动CPU消息应用程序。要实现这一步，选择一个在线的S7程序并且使用菜单命令**PLC > CPU Messages**为所选的CPU调用消息应用程序。

   结果：出现“CPU Messages”应用窗口，在此列出了所记录的CPU。

2. 可以通过重复步骤1.为其它程序或接口扩展注册CPU列表。
3. 在列表选项前点击复选框，并指定该模板应接收哪些消息：

   A：激活ALARM_S块的消息（SFC18和SFC108生成与块相关的消息总是需要确认，SFC17和SFC107生成与块相关的消息可以被确认），例如，从S7 PDIAG、S7-GRAPH或系统错误中报告过程诊断消息。

   W：激活诊断事件。

4. 设置档案的容量。

   结果：只要上述消息一出现，它们就被写入消息档案并且以所选择的方式显示。

注意

在SIMATIC Manager中使用菜单命令**PLC > CPU Messages**调用的CPU在“CPU **Messages**”应用窗口中已被存入注册模板列表。列表中的各项除非在“CPU Messages”应用窗口中删除，否则会一直保留下来。

16.8.2 显示存储的CPU消息

CPU消息总会被记录归档，除非选择了菜单命令“View > Ignore Message”。所有归档的消息总是会被显示。

16.9 组态“系统错误报告”

简介

当系统错误时，S7组件和DP标准从站（GSD文件决定从站的属性）可以触发组织块调用。

例如：如果有断线，具有诊断功能的模块就可以触发诊断中断（OB82）。

S7组件可以提供系统错误信息。开始事件信息，即赋值OB的局域数据（包括数据记录0）提供有关错误的位置（例如模块的逻辑地址）和类型（例如通道错误或备份错误）的一般信息。

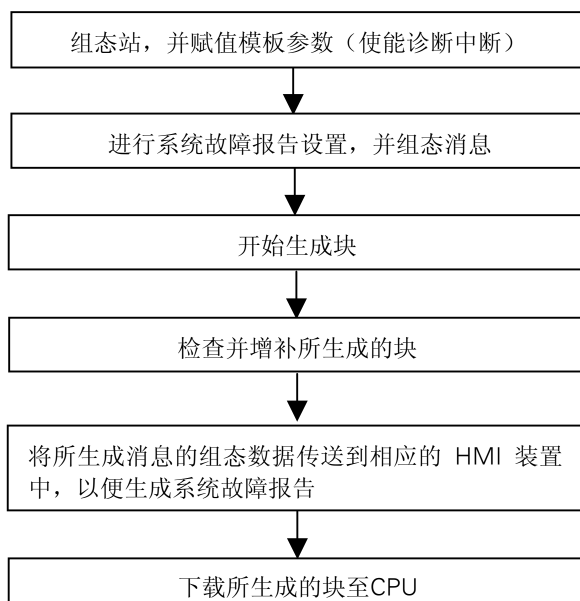
此外，通过附加的诊断信息错误可以被详述（使用SFC51读取数据记录1或使用SFC13读取DP标准从站的诊断消息）。例如通道0或通道1以及断线或测量范围超限错误。

使用“Report System Error（报告系统错误）”功能，STEP 7 提供了一种极其方便的方法显示由组件提供的诊断信息，以消息的形式显示。

STEP7可以自动生成所需块和消息文本。所有用户所需要做的就是将所生成的块装入CPU中，并将文本传送至所连接的HMI设备。

可在 “Supported Components（支持组件）”和“Functional Scope（功能范围）”部分中总览各种从站支持的的诊断信息。

基本步骤



通过ALARM_S/SQ标准消息路径，将消息传送至编程器上的CPU Messages中或所连接的HMI装置。

16.9.1 支持的组件和功能范围

如果S7-300站、S7-400站、DP从站和WinAC支持诊断、中断、插/拔模板中断以及通道诊断等功能，同样也支持“Report System Error（报告系统错误）”。

下述组件**不被**“Report System Error（报告系统错误）”所支持：

- M7、C7和在S7-300站中用于DP主站接口模板（CP 342-5 DP）。

在重新启动时，必须注意可能会出现丢失的中断信息。这是由于CPU的消息确认存储器在重新启动时，没有删除，但Report System Error 复位所有内部数据。

在下面两个表中，可找到Report System Error所支持的各种从站的所有诊断块。

诊断块	**ID(故障槽)**	**通道名称 (通道错误)**[1]	**模板状态** (模板错误、不正确/没有模板)	**设备名(只适用于ET 200 B)**
Header ID [2]	0x01	0x10	0x00 Type 0x82	0x00 + 1Byte Diagnostic Info
ET 200S	消息：“诊断可用”	纯文本消息	纯文本消息	- 4）
ET 200M	不评估	不评估	不评估	-
ET 200X	消息：“诊断可用”	-	-	-

诊断块	ID(故障槽)	通道名称 (通道错误)[1]	模板状态 (模板错误、不正确/没有模板)	设备名(只适用于ET 200 B)
ET 200X DESINA	消息："诊断可用"	纯文本消息	纯文本消息	-
ET 200L	不评估	-		-
数字ET 200B	消息："诊断可用"	-	-	消息："模板故障"
模拟ET 200B	消息："诊断可用"	-	-	
模拟ET 200C	消息："诊断可用"			消息："模板故障"
ET 200U	消息："诊断可用"			消息："模板故障"
ET 200iS	消息："诊断可用"	纯文本消息	纯文本消息	消息："模板故障"
ET200eco	-	-	-	纯文本消息
DP AS-i Link	消息："诊断可用"		纯文本消息	
ET200S(PN)	消息："诊断可用"	纯文本消息	纯文本消息	-
SCALANCE Switches	-	-	纯文本消息	-

诊断块	DS0/DS1 [1]	其它实例
Header ID [2]	0x00 Type 0x01	0x00 其它类型
ET 200S		
ET 200M	纯文本消息	不评估
ET 200X		
ET 200X DESINA	纯文本消息	
ET 200L	消息："模板故障"	
数字ET 200B		
模拟ET 200B	纯文本消息	
数字ET 200C		
模拟ET 200C	纯文本消息	
ET 200iS	纯文本消息	
ET200eco	-	-
DP AS-i Link	消息："模板故障"	
ET200S(PN)	-	-
SCALANCE Switches	纯文本消息	不评估

1）DS0：标准诊断，包含在OB82的局部数据中，长度4个字节。例如模板故障、外部辅助电源或前连接器丢失。
DS1：通道故障，各通道类型定义不一致，可通过SFC51从用户程序中读取。该文本来自S7 HW诊断。
2）头文件标识符：诊断消息的标识符，可以识别不同的诊断部分。

诊断消息（又叫作Norm从站消息）由上述诊断块组成，并可调用SFC 13在用户程序读取。

在STEP 7中，通过在线调用模块状态显示诊断消息。

诊断中继器：诊断中继器的消息以纯文本形式输出。消息读自GSD文件。

PROFINET

- 使用PROFINET IO，通道的诊断以纯文本形式输出。
- ET200S：从站支持地址打包方式。
- 使用PROFINET IO元件，支持生产厂商定义的诊断。

16.9.2 设置“Report System Error”

有几种方法可调用设置对话框：

- 在HW Config中，选择需要组态系统错误报告的CPU。选择菜单命令**Options > Report System Error**。
- 如果已经生成报告系统错误块，双击所生成的块（FB、DB），可以调用对话框。
- 在站的“Properties（属性）”对话框中，在Save（保存）和Compile（编译）组态时，选择自动调用选项。

在Save（保存）和Compile（编译）时，可以如下进入自动调用选项：

1. 在SIMATIC Manager中，选择相应的站。
2. 选择菜单命令**Edit＞Object Properties**。
3. 选择“Settings”选项标签。

注意：

可以通过菜单命令**Station > Properties**打开HW Config中的属性对话框中的“Settings”。

在对话框中，输入以下内容或其它选择：

- 生成FB以及分配的背景DB。
- 是否应生成参考数据。
- 是否在生成Report System Error（报告系统错误）时，总是显示警告。
- 是否在保存和编译组态后（见上述设置）自动调用Report System Error时，显示对话框。
- 生成错误OB：在S7程序中是否应生成仍不能使用的错误OB，在OB中“Report System Error”被调用。
- CPU出错特性：可以设置在系统出错报告后，是否将CPU切换为“STOP”。
- 是否确认消息。
- 消息外部特征（文本部分的结构和顺序）。
- 在用户程序块接口中包括哪些参数。

在调用对话框上的帮助中，会得到更详细的信息。

16.9.3 生成“Report System Error”块

当对报告系统出错的设定完成后，可以生成所需的块（FB和DB，并根据设置，包括并没有存在的DB块）。在“Report System Errors”对话框中点击“Generate”按钮。可以生成以下的块：

- 诊断FB（缺省：FB49）。
- 诊断FB的背景DB（缺省：DB49）。
- 错误OB（如果在“OB Configuration”对话框中进行了选择）。
- 由诊断FB调用的任选用户FB。

16.9.4 生成的FB、DB

所生成的FB可以评估故障OB的局域数据，读取触发故障的S7组件的其它诊断信息，并自动生成相应的消息。

FB具有以下特性：

- RSE（Report System Error，报告系统错误）生成语言（也适用于所生成的背景DB）。
- Know-how保护（也适用于所生成的背景DB）。
- 在运行期延时到达的中断。
- 双击调用“Report System Error”功能设置对话框（也适用于所生成的背景DB）。

用户块

由于诊断FB为know-how保护，因此不能对它进行编辑。但是，FB提供用户程序接口，因此可以存取错误状态或消息编号等。

在生成的FB中调用用户程序中的评估块（可以在对话框中的“User Block”选项标签中设置），生成的FB块带有选择的参数，下列参数是可用的：

名称	数据类型	注　释
EV_C	BOOL	// 进来（TRUE）或出去消息（FALSE）
EV_ID	DWORD	// 生成消息编号
IO_Flag	BYTE	// 输入模板：B#16#54 输出模板：B#16#55
logAdr	WORD	// 逻辑地址
TextListID	WORD	// 文本库的ID（缺省文本库= 1）
ErrorNo	WORD	// 生成故障编号
Channel_Error	BOOL	// 通道错误（TRUE）
ChannelNo	WORD	// 通道编号
ErrClass	WORD	// 错误等级
HerrClass	WORD	// H系统的错误等级

如果没有用户FB，可以使用上述参数由SFM生成。

标准故障所生成的错误文本排列如下：

错误号		影响的OB	OB中的错误代码	
从	至		从	至
1	86	OB 72	B#16#1	B#16#56
162	163	OB 70	B#16#A2	B#16#A3
193	194	OB 72	B#16#C1	B#16#C2
224		OB 73	B#16#E0	
289	307	OB 81	B#16#21	B#16#33
513	540	OB 82		
865	900	OB 83	B#16#61	B#16#84
1729	1763	OB 86	B#16#C1	B#16#C8

大于12288的错误号是指通道故障。如果以16进制显示错误号，则可以计算出通道类型和识别故障位。详细说明，参见相应模板的或通道的帮助文本。

例如：

12288 = W#16#3000 -> 高字节 0x30 – 0x10 = 通道类型 0x20 (CP接口)；

低字节 0x00，指错误位0

32774 = W#16#8006 -> 高字节 0x80 – 0x10 = 通道类型 0x70 (数字输入)；

低字节 0x06，指错误位6

16.9.5 在“Report System Error”生成外文消息文本

可以使用其它语言显示“Report System Error”中配置的消息，在安装STEP7时安装选择的语言。

如需要这样，按照下列步骤操作：

1. 在SIMATIC Manager中选择菜单命令Options >DisplayLanguage....。在弹出的对话框中添加需要的语言。
2. 点击OK键确认设定。
3. 在硬件配置中，选择菜单命令 Options >Report System Error....。在弹出的对话框中点击“Generate”按钮。

结果：生产所有语言的消息文本，但是这些文本只显示在“Add/Delete Language，Set Default Language”对话框中点击“Set Default Language ”按钮缺省设置的语言。

示例

在STEP7中安装德语、英语和法语，这些语言在项目中已经被定义。使用上述方法生成消息文本，为了使消息文本给定给的语言，在“Add/Delete Language，Set Default Language”对话框中将需要的语言设置为缺省。

17 控制和监视变量

17.1 组态操作员控制和监视变量

概述

当使用WinCC对过程或可编程控制器进行监控时，STEP 7可提供用户友好方式对变量进行控制和监视。

与前面的方法相比，这一方法的优势在于不需要再去给每一个操作站（OS）分别组态数据，只需要使用STEP 7作一次组态即可。可以使用传送程序AS-OS Engineering（“Process Control Syetem PCS 7”软件包的一部分）将STEP 7中组态生成的数据传送给WinCC，在传送过程中要检查数据的一致性及它们与显示系统的可兼容性。WinCC使用变量块及图象对象中的数据。

在STEP 7中，可以为下列变量组态或修改操作员控制和监视属性：

- 输入、输出及功能块的输入/输出参数
- 存储位及I/O信号
- CFC功能图中的CFC参数

基本步骤

组态操作员控制及监视变量的基本步骤要依据所选的编程/组态语言以及需要控制和监视的变量而定。但基本步骤总会包括以下各步：

1. 将操作员控制及监视的系统属性赋给功能块的参数或赋给符号表中的符号。

   在CFC中无需此步骤，因为可以使用库中已准备好的块。
2. 给需要控制和监视的变量赋予所需的属性，在属性对话框中选择控制属性（S7_m_c）。在“Operator Interface（操作员接口）”对话框中（菜单命令**Edit > Special Object Properties > Operator Interface**），可以改变变量在WinCC中的属性，例如极限值、替代值以及协议特性等。
3. 使用工具PLC-OS Engineering将STEP 7中生成的组态数据传送到显示系统（WinCC）。

命名规则

为WinCC存储和传送的组态数据要存储在由STEP 7自动赋给并且唯一的名字下面。用于操作员控制及监视、CFC功能图以及S7程序的变量名是组成该名字的一部分，因此要符合以下规律：

- 在S7项目下的S7程序的名字要唯一（不同的站不能包含同名的S7程序）。
- 变量、S7程序和CFC功能图的名字不能包含下划线、空格或下列特殊字符：['] [.] [%] [-] [/] [*] [+]。

17.2 用 STL、LAD 及 FBD 组态操作员控制及监视的属性

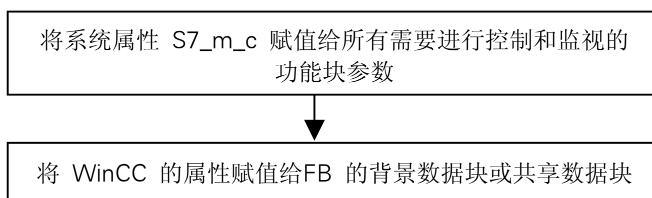
概述

使用下面描述的步骤，可以设置功能块的参数以适合于操作员控制和监视，并且可以将所需的O、C、M属性赋值给用户程序中相关的背景数据块或共享数据块。

要求

必须已经创建一个STEP 7项目、一个S7程序以及一个功能块。

基本步骤



17.3 用符号表组态操作员控制与监测属性

概述

不论使用什么编程语言，都可以按下面所描述的步骤组态下列变量：

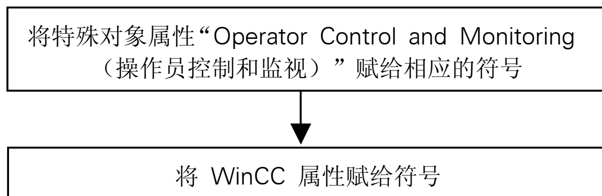
- 位存储
- I/O信号

要求

在开始之前，必须满足下列要求：

- 在SIMATIC管理器中已生成一个项目。
- 在该项目下有一个带有符号表的S7程序。
- 该符号表必须是打开的。

基本步骤



17.4 用 CFC 改变操作员控制和监视属性

概述

在CFC中，可以从库中选择已有操作员控制和监视功能的块来生成用户程序，并且将这些块在功能图相应的位置上进行连接。

要求

在STEP 7项目中已插入一个S7程序，在块中生成了CFC功能图。

基本步骤

编辑块的对象属性

注意

如果使用自己生成的块，并且已经分配了系统属性S7_m_c，可以通过激活“Operator Control and Monitoring”对话框中的“Operator Control and Monitoring”复选窗口将这些块赋予操作员控制和监视能力（菜单命令**Edit > Special Object Properties > Operator Control and Monitoring**）。

17.5 传送可编程控制器组态数据到操作员接口

概述

使用传送应用AS-OS Engineering，可以将控制和监视数据传送到WinCC的数据库中。

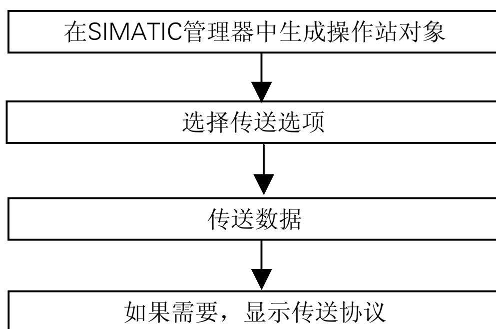
要求

在开始传送之前，下列要求必须满足：

- 已经安装了AS-OS Engineering。
- 已为操作员控制和监视生成了组态数据。

基本步骤

将操作员控制和监视的组态数据传送到WinCC数据库，其过程如下：



18 建立在线连接并进行CPU设置

18.1 建立在线连接

在编程器和可编程控制器之间建立在线连接，用于下载S7用户程序/块、从S7可编程控制器上传程序到编程器和其它操作：

- 调试用户程序
- 显示和改变CPU操作模式
- 为CPU设置时间和日期
- 显示模板信息
- 比较在线和离线的块
- 诊断硬件

为建立在线连接，编程器和可编程控制器必须通过合适的接口相连接（例如，多点接口（MPI））。然后，可以通过一个在线的项目窗口或“Accessible Nodes（可访问站点）”窗口访问可编程控制器。

18.1.1 通过“Accessible Nodes”窗口建立在线连接

这种访问方式可以快速访问可编程控制器，例如用于测试的目的。可以在网络中访问所有的可以访问的可编程控制器。在编程器中没有可编程控制器的项目数据时可选择该种方式。

使用菜单命令**PLC > Display Accessible Nodes**，打开“Accessible Nodes”窗口。在“Accessible Nodes”对象中显示网络中所有可访问的可编程控制器站点及其地址。

那些不能用STEP 7编程的站（如编程器或操作面板）也能显示出来。

在圆括号中还显示了下列信息：

- (direct)：该节点直接连接到编程设备上(编程器或PC机)
- (passive)：该站点不能通过PROFIBUS DP进行编程和状态监控或修改
- (waiting)：由于该节点与网络中其它的设置不匹配，所以不能与该节点通信

寻找直接连接的站点

附加信息“direct”不支持PROFInet站点。为了可以寻找直接连接的节点，选择**PLC>Diagostics/Settings>Node Flashing Text**菜单命令。

在显示的对话框中可以设定闪烁时间并且可以开始闪烁测试。闪烁的FORCE LED将指示直接连接的节点。

FORCE功能激活状态下不能执行闪烁测试功能。

18.1.2 通过在线的项目窗口建立在线连接

如果在编程器/PC的项目中有组态的可编程控制器，可以选择该种方法。在SIMATIC Manager中使用菜单命令**View > Online**打开一个在线窗口。该窗口显示可编程控制器中的项目数据（于离线窗口相比较，离线窗口显示编程器/PC中的项目数据）。可编程控制器中的S7程序和M7程序的数据都可以在在线窗口中显示。

可以将项目的这种显示用于与访问可编程控制器相关的功能。SIMATIC Manager中，菜单“PLC”中的某些功能只能在在线窗口中激活，不能在离线窗口中使用。

有以下两种访问类型：

- **带有组态硬件的访问**
  这意味着只能访问离线组态的模板。哪些模板可以在线访问由组态可编程模板时设置的MPI地址决定。
- **不带有组态硬件的访问**
  这种方法要求有一个与硬件无关的S7程序或M7程序存在（即该程序直接位于项目之下）。由S7/M7程序对象属性中定义的相应的MPI地址决定哪个模板在线可以访问。

通过在线窗口访问，显示的数据是可编程控制器与编程器中组合的数据。例如，如果在一个在线项目下打开一个S7块，显示由下列组合：

- 来自于可编程控制器CPU中的块指令代码部分，及
- 来自编程器数据库中的注释和符号（如果离线程序中有这些注释和符号）。如果不通过项目结构，而是直接打开所连的CPU，则显示的程序就是这些程序在CPU中的原样，即没有符号和注释。

18.1.3 在多重化项目中在线访问PLC

用一个PG/PC进行项目间访问

用于对象“PG/PC”和“SIMATIC PC站”的“Assign PG/PC”功能同样适用于多重化项目中。

可以在多重化项目中任何一个项目中指定需要在线访问的目标模板。其步骤与仅在一个项目中进行访问是一样的。

需求

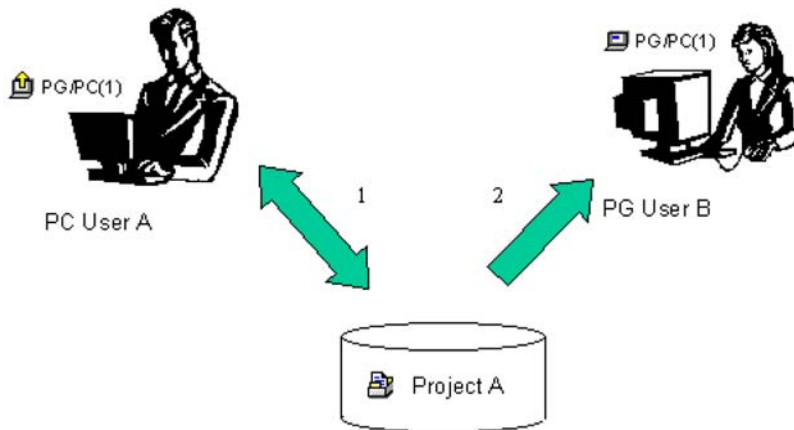
- 用于在线访问PLC的PG/PC或PC站必须已经在多重化项目中的任何一个项目中进行了分配。注意：当相应的项目打开时，所分配的PG/PC或PC站以黄色高亮显示。
- 所有项目的子网已连接。
- 所有项目已进行了编译，组态数据已下载到相应的站；例如，为建立连接的PG/PC和目标模板间提供路由信息。
- 通过网络可以访问目标模板。

当工作在分布式项目中可能出现的问题

如果所分配的项目已经改变以及在PG/PC上打开一个还没有生成的项目时，PG/PC的分配是不可见的。

即使如此，所组态的PG/PC对象还依然保持所分配的状态，但是PG/PC分配是错误的。

在这种情况下，必须清除现有的分配并重新分配PG/PC对象。这样可以没有问题地在多重化项目中在线访问所需的站点。



1. 在网络中用所分配的PG/PC保存项目A
2. 用不同的计算机打开相同的项目A

在分布式项目中的应用窍门

如果一个工作组中多个人要访问与其PG上连接的PLC，最好在多重化项目中建立一个“PG/PC”或“SIMATIC PC站”，然后再为每个PG建立一个连接分配。

取决于打开了项目的PG，SIMATIC　Manager只以黄色箭头形式指示分配给该PG的对象。

18.1.4　设定访问可编程控制器的口令

使用口令保护，可以：

- 保持CPU中的用户程序及其数据未经授权不能改变（写保护）
- 保护用户程序中的编程专利（读保护）
- 保护在线功能以免可能对过程造成干扰

如果模板支持口令保护功能，就可以保护模板或MMC（例如CPU31XC）的内容。

如果需要口令来保护一个模板或MMC的内容，则必须在对模板进行参数赋值的过程中定义保护级别并设置口令，然后将修改的参数下传到模板。

如果需要输入口令执行在线功能或或MMC的内容，“Enter Password”对话框会显示出来。若输入的口令正确，就可以得到模板的访问权，该模板在参数赋值过程中已被设置了特定的保护级别。此时你就可以与被保护的模板建立在线连接并执行属于那个保护级别的在线功能。

使用菜单命令**PLC>Access Rights> Setup**，可以直接调用“Enter Password”对话框。在线访问开始时，可以只输入一次口令，以后在线操作时，将不再询问。输入的口令将有效至SIMATIC Manager关闭或使用菜单命令**PLC > Access Rights > Cancel**取消口令。

CPU参数	要点
测试操作/过程操作（不适用于S7-400 或CPU 318-2）	可在“Protection”标签中设置 在过程操作中，为保证设置允许的循环扫描时间不超限，程序测试或监视 / 修改变量这样的测试功能将受到限制。程序状态和状态显示的调用被限制，可能造成程序调用的中断。 使用断点的测试及程序的单步执行在过程操作中不能使用。 在测试操作中，通过编程器/PC的所有测试功能都可以不受限制地使用，即使这些功能可能会使循环扫描时间的加大增加。
保护级别	可在“Protection”标签中设置，如果知道正确的口令，就可以对CPU进行读/写访问。口令在此标签中设定。

18.1.5 刷新窗口内容

需要注意以下事项：

- 由于用户操作（如下载或删除块）而在一个在线的项目窗口进行的修改不会在任何已打开的“Accessible Nodes（可访问站）”窗口自动刷新。
- 任何在“Accessible Nodes”窗口中所作的修改，不会在任何已打开的在线项目窗口中自动刷新。

要刷新一个并行打开的窗口，必须直接在该窗口刷新（使用菜单命令或功能键F5）。

18.2 显示和改变操作模式

18.2.1 显示和改变操作模式

例如用这个功能，例如可以在纠正错误之后将CPU切换到“RUN”模式。

显示操作模式

1. 打开项目并选择S7/M7程序，或用菜单命令**PLC > Display Accessible Nodes**打开“Accessible Nodes”窗口并选择一个站（“MPI=…”）。
2. 选择菜单命令**PLC > Diagnostics/Settings > Operating Mode**。

该对话框显示当前和最近一次的操作模式以及在模板上当前模式选择开关的设置。如果无法显示其当前开关设置的模板，则显示文本“Undefined”。

改变操作模式

可以使用按钮改变CPU的模式。只有当这些按钮是激活的，才能在当前的操作模式下进行选择。

18.3 显示并设置时间和日期

18.3.1 带有时区设定和夏令/冬令时的CPU时钟

处日期时间外，还可以用STEP 7 V5.1 SP2在新的CPU中(固件V3以上)设置或评估下列设定

- 夏令/冬令时
- 时区偏移系数

时区显示

系统通过TOD运行。TOD是一个全球的、连续的、不中断的模板时间。

本地自动化系统允许计算与模板时间不同的当地时间，并可在用户程序中使用。当地时间不直接输入，它是通过模板时间加/减与模板时间的时差来计算的。

夏令时/冬令时

当建立了TOD和日期后也可以设置夏令时或冬令时。当从夏令时切换到冬令时时，用户程序只需考虑与模板时间的时差。

读取和调整TOD以及TOD状态

夏令时/冬令时的标识以及与模板时间的时差包含在TOD的状态中。可通过下列方法读取或调整TOD及其状态：

使用STEP 7(在线)

- 通过菜单命令**PLC > Diagnostics/Setting > Adjust TOD**(读取和调整)
- 通过“模板信息”对话框中的“Time System”(只读)

在用户程序中

- SFC 100“SET CLKS”(读取和调整)
- SFC 51“RDSYSST” with SZL 132, Index 8(只读)

诊断缓冲区、消息和OB启动信息中的时间标签

用模板时间生成时间标签。

TOD中断

当冬令时切换到夏令时时，由于“时间跳转”而没有触发TOD中断，则调用OB 80。

冬令时/夏令时的转换不会影响TOD中断的触发周期，例如每分钟、每小时触发TOD中断。

TOD同步

设置为主TOD的CPU(例如在CPU属性“Diagnostics/Clock”中设置)总是同步其他时钟的模板时间和当前TOD状态。

18.4 更新固件

18.4.1 在线更新模板和子模板中的固件

从STEP 7 V5.1 SP3以后，可以使用标准方法更新一个站中的模板或子模板的固件。步骤如下：

概念

更新诸如一个CPU、一个CP或一个IM模板的固件时，必须有包含最新固件的信息文件(*.UPD)。

选择一个文件并下载到模板中。

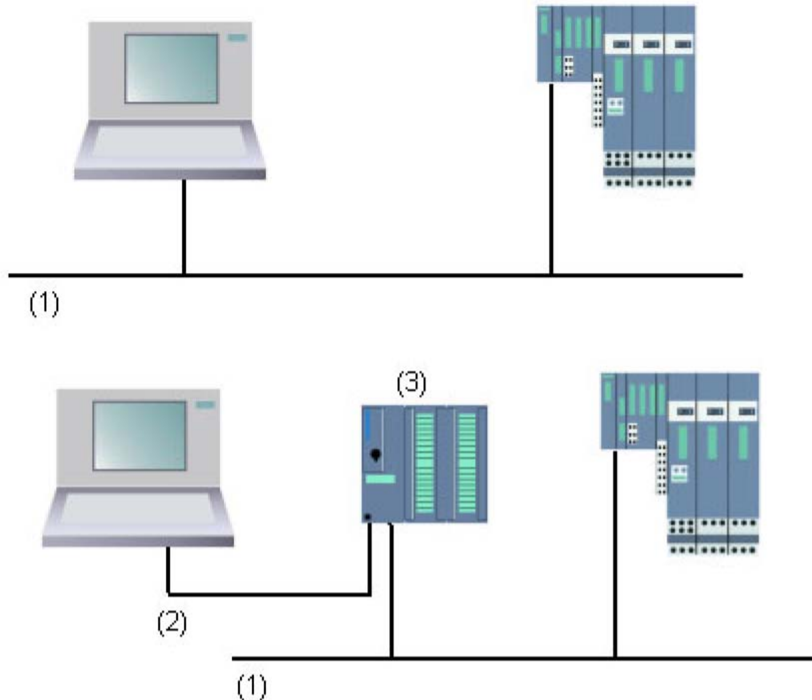
前提条件

要更新固件的模板必须能够在线访问。编程器必须与要更新固件的模板在一个MPI、PROFIBUS或以太网中。当编程器连接到DP主站CPU的MPI接口并且需要更新固件的模板

通过DP接口连接到的PROFIBUS网络或通过PN接口连接到以太网上，也可以更新固件。但是CPU必须支持MPI接口与DP接口或MPI接口与PN接口的S7路由。

此外，该模板必须也支持固件更新。

PG/PC机上的文件系统中必须包含最新固件版本的文件。一个固件版本的文件必须在一个文件夹中。



(1) PROFIBUS 子网
(2) MPI 子网
(3) CPU带有MPI接口和DP接口(带S7路由)

在硬件配置中的更新步骤

1. 打开需要更新模板的站。
2. 选择模板。对于类似IM 151 PROFIBUS DP接口模板，为DP从站选择图标。在这种情况下，它是一个标准的ET 200S。
3. 选择菜单命令**PLC > Update firmware**。如果所选择的模板或DP从站支持“更新固件”功能，菜单命令才被激活。
4. 在显示的“更新固件”对话框中点击“Browse”按钮并选择固件更新文件(*.UPD)路径。
5. 选择完一个文件后，下方的“更新固件”对话框中将告诉该文件适用于哪个模板以及其固件版本。

6. 点击“RUN”按钮。STEP 7将检查所选择的文件是否适用于该模板，如果正确，则向模板下载该文件。如果需要改变CPU的运行模式，对话框将提示执行这些步骤。

   此时模板将独立进行固件更新。

注意：对诸如CPU 317-2 PN/DP进行固件更新时，通常需要与CPU建立一个单独的连接。在这种情况下可以对过程进行中断。如果其它连接没有连接源，空闲的通信连接被自动使用，此时连接不能被中断。此时传送对话框中的“Cancel”按钮变为灰色，并且不能使用。

7. 在STEP 7中检查(读CPU诊断缓冲区)是否能使用新固件启动该模板。

在SIMATIC Manager的更新步骤

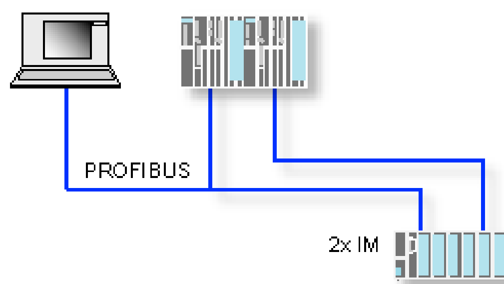
只有在“Accessible Nodes”窗口打开的情况下才能激活此功能，与在硬件配置中的更新相同，也是选择菜单命令**PLC > Update firmware**进行更新。**STEP7**将检查模块是否支持固件更新功能。

在冗余的模式下更新模块的固件

从STEP7 V5.4以后，支持模块在冗余模式下的固件更新。例如在H系统中更新PROFIBUS-DP的接口模块 IM 153-2BA00。可以在单个接口模块工作的模式下对冗余的接口模块逐个进行固件更新，最新的固件版本将下传到冗余的接口模块中。

要求：编程器PG或PC必须连接到与接口模块相同的PROFIBUS网络上，在SIMATIC Manager “Accessible Nodes”窗口中执行更新任务。

冗余模式更新固件示意图：



运行期间更新固件的顺序

可以在更新对话框中选择是否更新后直接激活新固件。

如果选择该选项后，相应站将执行类似于关电/上电的重启动。可以在CPU处于STOP模式或对用户程序处理没有影响的情况下进行，在运行中更新固件应采取相应的预防措施。

例如会发生站中所有模板包括已有的F I/O出现重启动失败现象。

当发生电源调电或切换到故障安全状态，F I/O模块接口输出通信故障并被钝化，在重新启动后，F I/O模块接口钝化不能被清除，必须单独为模块去钝化，这样故障安全的应用将不能直接启动。

19 下载和上传

19.1 从 PG/PC 中下载到可编程序控制器

19.1.1 下传条件

下传到可编程序控制器的条件

- 编程器和可编程序控制器的CPU之间必须有一个连接（例如，多点接口）。
- 必须可以访问可编程序控制器。
- 向PLC下传块时，在项目的对象属性对话框中，在“Use”选项中选择“STEP 7”。
- 下传的程序已无误地编译。
- CPU必须在允许下传的工作模式下（STOP或RUN-P）。
  RUN-P模式表示，这个程序将一次下传一个块。如果覆盖一个旧的CPU程序，可能出现冲突，例如，块参数已改变了。当循环处理时，CPU就会进入STOP模式。因此建议在下传前将CPU切换到STOP模式。
- 如果需要离线打开一个块并要下传它，CPU必须与SIMATIC Manager中的一个在线用户程序相连接。
- 在下传用户程序之前，必须复位CPU，保证没有旧的程序块保留在CPU上。

STOP 方式

在做下列事情前，把操作模式从“RUN”模式切换到“STOP”模式：

- 下传完整或部分用户程序到CPU
- 在CPU中复位所有的存储器
- 压缩用户存储器

暖启动（切换到“RUN”模式）

如果在“STOP”模式时执行一个暖启动，程序重新启动并在“STARTUP”模式下首先运行启动程序（在块OB100里）。如果启动成功，CPU转到“RUN”方式。在下列情况下需要暖启动：

- CPU复位
- 在STOP模式下传用户程序

19.1.2 保存和下传块的区别

一定要区分块的保存和下传。

	保　存	下　传
菜单命令	File > Save File > Save As	PLC > Download
功能	编辑器中块的当前状态被保存在编程器的硬盘上	编辑器里块的当前状态仅下传到CPU上
语法检查	运行语法检查。任何错误都将在对话框里被报告。错误的原因和位置也将显示出来。在下传或保存块前必须纠正这些错误。如果在语法上没有发现错误，块将被编译成机器码并且保存或下传。	运行语法检查。任何错误都将在对话框里被报告。错误的原因和位置也将显示出来。在下传或保存块前必须纠正这些错误。如果在语法上没有发现错误，块将被编译成机器码并且保存或下传。

这个表的应用与在线或离线打开块无关。

块修改的技巧——先保存后下传

要存入新创建的块或在逻辑块的程序区中进行修改，在声明表中修改或在数据块中输入新的或更改的数据值，必须对各个块进行保存。用命令菜单PLC > Download可将在编辑器里进行的任何修改传送到CPU，例如，检查小的修改，在退出编辑程序前，任何情况下都必须（把变更）保存在编程器的硬盘上，否则，在CPU中与编程器上将得到不同版本的用户程序。一般建议先保存所有的更改，然后再下传。

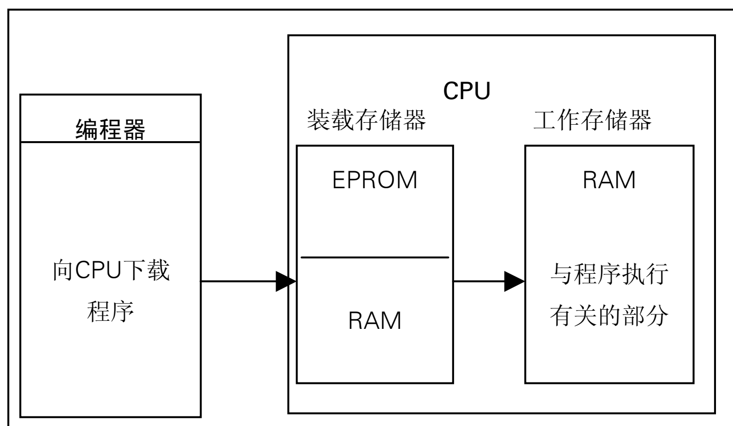
19.1.3 CPU里的装载存储器和工作存储器

在完成组态、参数赋值、程序创建和建立在线连接后，可以下传整个用户程序或个别块到一个可编程序控制器。要测试单个块，必须至少下传一个组织块（OB），在OB里调用功能块（FB）和功能（FC）以及使用的数据块（DB）。要下传硬件组态、网络组态和创建连接表生成的系统数据到可编程控制器，可选择下传对象“System Data”。

在程序测试的末段或运行最终用户程序期间，使用SIMATIC　Manager将用户程序下传到可编程序控制器中。

装载存储器和工作存储器之间的关系

完整的用户程序下载到装载存储器，与程序执行有关的部分装入到工作存储器中。



CPU 装载存储器

- 装载存储器用来存储没有符号表和注释的用户程序(这些符号和注释保留在编程器的硬盘中)
- 没有执行的块，将仅存储在装载存储器中
- 装载存储器可以是RAM，ROM或EPROM存储器，随可编程序控制器而定
- 装载存储器可以是集成的EEPROM或集成的RAM（如，CPU 312 IFM和CPU 314 IFM）
- 在S7-400中，使用存储卡（RAM或EEPROM）可用扩展装载存储器

CPU工作存储器

工作存储器（集成RAM）用来存储程序处理所需要的那一部分用户程序。

可能的下传/上载步骤

- 用下传功能下传用户程序或可装载的对象（如，块）到可编程序控制器。如果这个块已存在于CPU的RAM中，将提示是否需要覆盖这个块。
- 可以在项目窗口中选择可装载的对象，并从SIMATIC Manager中下传它们（菜单命令：PLC > Download）。
- 当编写程序块、组态硬件和网络时，可直接下传当前正在编辑的对象，可以使用主窗口里的菜单下传程序。（菜单命令：PLC > Download）。
- 打开一个在线窗口查看可编程控制器（例如，用View > Online或PLC > Display Accessible Nodes），将需要下传的对象复制到在线窗口中。

另外可以使用上载功能将CPU装载存储器RAM中的程序块上载到可编程序控制器中。

19.1.4 下载方式随装载存储器而定

CPU装载存储器中使用的RAM和EEPROM决定下传用户程序或块时可用的方式。以下是下载数据到CPU中的可能方式：

装载存储器	下传方式	PG和PLC之间的通信类型
RAM	下载和删除各个块	PG-PLC在线连接
	下载和删除整个用户程序	PG-PLC在线连接
	重新装入独立的块	PG-PLC在线连接
集成（仅S7-300）或外插EPROM	下载整个用户程序	PG-PLC在线连接
外插EPROM	下载整个用户程序	通过EPROM适配器或EPROM插入PLC中，使用CPU接口写入程序

通过在线连接下传到RAM

在可编程序控制器中，如果电源故障或RAM没有被备份时则数据丢失。这种情况下，RAM中的数据也将随着丢失。

保存到EPROM存储卡中

块或用户程序被保存在一个EPROM存储器上，然后插在CPU的一个插槽中。

存储卡是便携式数据记录媒体。它们通过编程器来写入，然后插在CPU上恰当的插槽里。

当电源关断后和CPU复位时，保存在它们上面的数据将被保留。在CPU存储器复位且电源掉电之后，电源重新恢复时如果RAM中的内容没有备份，EPROM中的内容被重新复制到CPU存储器的RAM区。

保存在集成EPROM里

对于CPU312，也可以保存RAM的内容到集成EPROM中。在电源断开时，集成的EPROM中的数据将被保留。在电源断开且CPU存储器复位后再重新恢复时，如果RAM没有被备份，集成的EPROM中的内容将被重新复制到CPU存储器的RAM区。

19.1.5 下传程序到S7 CPU

19.1.5.1 使用项目管理器下载

1. 在项目窗口中选择要下载的用户程序和程序块。
2. 通过菜单命令**PLC>Download**将所选择的对象下载到可编程序控制器中。

另一种方法 – Drag & Drop (脱放功能)

1. 打开一个离线窗口和一个在线窗口。
2. 将离线窗口中要下载的对象脱放到在线窗口中。

19.1.5.2 不使用项目管理器下载

1. 用菜单命令**PLC>Display Accessible Nodes**或点击工具栏中的相应按钮来打开“Accessible Nodes”。
2. 在所需要的站点上双击“Accessible Nodes”来显示“Blocks”对话框。
3. 用菜单命令**File>Open**打开用户程序中要下载的项目或者要向PLC下载的块。
4. 在打开的项目窗口中选择要下载的对象。
5. 用脱放功能将向PLC下载的对象复制到“Accessible Nodes”中的“Blocks”文件夹中。

19.1.5.3 重新向PLC中下载块

可以使用新的程序块覆盖S7 PLC中工作存储器或装载存储器中存在的程序块(重新下载)。

块的重新下载和下载具有相同的步骤。重新下载时会提示是否要覆盖已有的块。

不能删除EPROM中存储的块，只是在重新下载时被声明一次。其替换的块将下载到RAM中，这将导致在工作存储器或和装载存储器中产生间隙，这些间隙将最终导致无法再下载新块，这时需要对存储器进行压缩。

注意

如果断电后并再次上电，并且RAM没有后备电池，或者CPU的存储器复位，则以前的“旧”块将不再有效。

19.1.5.4 将下载的程序块保存到集成EPROM中

对于具有集成EPROM的CPU(例如CPU 312)，可以将RAM中的块保存到集成的EPROM中，这样即使掉电或存储器复位时也不会丢失数据。

1. 使用菜单命令**View>Online**显示一个被打开项目的在线视窗，或点击工具条中的“Accessible Nodes”按钮打开“Accessible Nodes”视窗，也可通过菜单命令**PLC>Display Accessible Nodes**打开该视窗。
2. 选择项目在线视窗中的S7或M7程序，或“Accessible Nodes”视窗中的节点。
3. 使用下列一种方法选择CPU中的“Blocks”文件夹，将向该文件夹中存储块。
   - 如果使用项目管理器，则在项目的在线视窗中选择
   - 如果不使用项目管理器，则在视窗的“Accessible Nodes”中选择
4. 选择菜单命令**PLC>Save RAM to ROM**。

19.1.5.5 通过EPROM存储卡下载

要求

使用编程器访问适用于S7 PLC的EPROM存储卡必须有合适的ERPOM驱动器。访问适用于M7控制系统的EPROM存储卡，则必须安装了FLASH文件系统（只能在PG720、PG740和PG760上）。当安装STEP 7标准软件包时，EPROM驱动器和FALSH文件系统作为可选项提供。如果使用PC，要存储到EPROM存储卡需要一个外置EPROM写入装置，也可以在稍后安装驱动器。如果需要安装驱动，通过菜单命令**Start>Simatic>STEP 7>Memory Card Parameter Assignment**调用相应的对话框，或者通过控制面板（双击图标“Memory Card Parameter Assignment（存储卡参数赋值）”）。

保存在存储卡上

要将块或用户程序保存到存储卡可按如下进行：

1. 在编程器的槽口中插入存储卡。
2. 打开“Memory Card（存储卡）”窗口，可用以下方式：
   - 点击“Memory Card”按钮。如有必要可用菜单命令**View > Toolbar**激活工具栏。
   - 选择菜单命令**File>S7 Memery Card>Open**。
3. 打开或激活需要存储程序块的下列窗口之一：下列窗口都可以：
   - 项目窗口，“在线”视窗
   - 项目窗口，“离线”视窗
   - 库窗口
   - “Accessible Nodes（可访问站）”窗口
4. 选择“Blocks”文件夹或单个块并将它们拷备到“S7 Memory Card”窗口中。
5. 如果存储卡中已有一个块，则显示错误信息。这种情况下，删除存储卡中的内容然后从步骤2开始重复。

19.2 在 PG 上编译并下载多个对象

19.2.1 下载的条件和注意事项

下载块文件夹

只能下载块文件夹中的逻辑块，块中的其它对象诸如系统数据块(SDB)等不能被下载。SDB只能通过“Hardware”对象进行下载。

注意

使用“Compile and Download Objects”不能下载PCS 7项目中的块，只能通过CFC下载以保证正确的下载顺序。这样做才能防止CPU进入STOP模式。

检查项目属性可以决定该项目是否是PCS 7项目。

故障安全型控制器F-Share的下载

出于安全考虑，下载修改的F-share前必须输入一个密码。因此当使用“Compile and Download Objects”功能时，将会终止下载过程并发出一个错误信息。在这种情况下，将程序的相应部分连同选项包一起装载到PLC中。

下载硬件配置

使用“Compile and Download Objects”功能下载硬件配置(例如下载离线SDB)，如果没有出现错误信息或提示，下载过程将中断地执行。下面章节将介绍如何避免出现这样的错误信息或提示。

下载硬件配置的条件

- CPU必须处于STOP模式。
- 必须与CPU建立在线连接。如果所选择的CPU或所选择的块文件夹具有口令保护，则在“Compile and Download Objects”功能运行前需要一个授权的连接或输入口令(“Edit”按钮)。
- 目标系统的下载接口不能被重新组态：
  - 不能改变接口地址
  - 如果改变网络设置，将会导致有些模板不能被访问
- 对于H-CPU，在运行“Compile and Download Objects”功能前(选择“CPU”对象并点击“Edit”按钮)，选择需要下载的CPU(H-CPU0或H-CPU1)。
- 不能修改下列CPU参数：
  - 局部数据最大容量和CPU上的通讯资源
  - F-CPU的口令保护
- 对于每个组态的模块，必须满足下列条件：
  - 所组态模板的订货号必须与实际插入模板的订货号一致。
  - 所组态模板的固件号不能高于实际插入模板的固件号。
  - 从最后一次下载后，项目名称、站名和模板名称都不能改变，但是可用分配新的项目名称。

下载过程的一些技巧

- 所有离线的SDB将被下载(也就是说除了硬件配置，还有连接的SDB和通过全局数据组态创建的SDB)。
- 当编译正确无误时才能下载。

- 下载期间，将抑制任何错误的反馈信息。例如如果出现CPU存储瓶颈，在不通知用户的情况下自动对数据进行压缩。
- 下载完成后，所下载的模块将进入STOP模式(除了那些可以自动停止和从新启动并没有用户提示的模块)。

窍门

下载完成后，如果出现警告提示，则必须要查看错误日志中的内容，此时可能对象没有下载或没有全部下载。

19.2.2 如何编译和下载对象

在“Compile and download objects”对话中选择一个项目或多重化项目中要传送到PLC的对象并且选择下载顺序(如果需要的话)。该对象用于传送一个站、一个项目或多重化项目。

根据所选的对象不同，可能不会出现确切的信息。此外，可能不是下面所描述的所有功能都适用于这些对象，特别是用选项软件包生成的对象。

对于一个块文件夹中的块，“编译”将意味着对块进行一致性检查。

步骤：

1. 在SIMATIC管理器中，选择要编译或者编译后下载的块。在SIMATIC Manager中可以选择以下的对象：
   - 多重化项目
   - 单项目
   - 站
   - 没有分配站的S7程序
2. 在SIMATIC管理其中选择菜单命令**PLC>Compile And Download Objects**。
3. 如果只想检查块而不想下载到PLC时，选择“Only compile”。
4. 当编译出错时为了防止将其下载到PLC中，应选择“No download on compilation error”。选择该选项后，将不会下载任何对象。如果没有选择，则所有编译正确的对象将下载到PLC中，而编译出现错误的块将不会被下载。
5. 如果想同时完成编译和下载连接，为“connections”选择相应的复选框。
6. 多重化项目特别适合作为下载的开始点，项目中包括所有与之相连的站点和站点间的通信连接。
7. 在“编译”和“下载”栏中，选择要编译或下载的对象，选择后将出现选择标记。如果在第3步选择了“Compile only”，则“下载”栏将显示为灰色并且不能使用。
8. 点击“Start”按钮开始编译。
9. 按照屏幕上显示的指令完成下载和编译。

在编译和下载完成后，显示一个完整的日志，可以打开所有日志或选择单一对象日志：

- 点击“All”按钮查看所有错误日志。
- 点击“Single object”查看从对象表中所选择对象的错误日志。

编译下载连接时特殊的考虑

选择“Connection”作为编译对象，STEP7自动选择相应“Connection” 对象的通信方。经过编译，STEP7生成一致的组态数据（系统数据块）。自动选择的对象不能通过手动取消选定，如果最初选择的“Connection”被取消选定，选择将自动去除。

选择“Connection”作为下载对象，STEP7自动选择“Compile”复选框，此外，STEP7为所有连接的通信方选择“Compile”和“Download”复选框。如果只有“Connection”被选中，也可以在CPU “RUN-P”的操作模式下载。

可用使用Netpro下载独立的连接。

编译下载硬件：连接的影响

选择“Hardware”作为编译和下载对象，所有硬件下的连接被自动选择进行编译和下载。在这种情况下，连接的通信方不能被下载下载。

19.3 从可编程控制器上传到 PG/PC

当执行以下操作时，上传功能支持：

- 保存来自可编程序控制器中的信息（例如，为了维护目的）
- 快速组态和编辑一个站，如果在开始组态前硬件部分是可用的

保存来自可编程序控制器的信息

这个措施是必需的，例如，如果这个版本的离线项目数据不能或部分不能在CPU上运行。这种情况下，至少可以得到在线可用的项目数据并上传它们到编程器上。

快速组态

如果组态完硬件并重新启动站后，从可编程序控制器中上传组态数据到编程器，那么站组态就比较容易。这站组态和每个模块的类型将被提供。然后要做的事情就是以更多的细节来定义这些模块（订货号），并且对它们进行参数赋值。

下列信息可上传到编程器上：

- S7-300：中央机架和任何扩展机架的组态
- S7-400：带一个CPU和信号模块的中央机架的组态，没有扩展机架
- 分布式I/O的组态数据不能上传到编程器上

如果在可编程序控制器上没有组态信息，则这个信息将被上传；例如，如果在系统上执行了一个存储器的复位。否则，**上传**功能可提供更好的结果。

对于没有分布式I/O的S7-300系统，需要做的全部事情就是以更多的细节来定义这些模块（定化号），并且对它们进行参数赋值。

注意

当上传数据时（如果没有一个离线组态），STEP 7不能确定全部组件的订货号。

当用命令菜单Options>Specify Module组态硬件时，可以输入“未完成的”订货号。这样，可以对STEP 7不能识别的模块进行参数赋值（那是指，没有出现在“硬件样本”窗口里的模块）；不管怎样，STEP 7将不会检查是否遵守参数法则。

从可编程序控制器上传时的限制

下列限制适用于从可编程序控制器上传到编程器的数据：

- 块不包含任何参数、变量和标号的符号名称
- 块不包含任何注释
- 上传带有全部系统数据的完整的程序，只有这样系统才能继续处理属于“组态硬件”应用程序的系统数据
- 不能更进一步地处理全局数据通信（GD）和组态符号相关消息
- 强制作业不能把其他数据一起上传到编程器上。它们必须作为变量表（VAT）被单独保存
- 模块对话框里的注释不能上传
- 只有在组态期间，这个选项被选中，模块的名称才会被显示（HWConfig：在Options>Customize对话框中的 “在可编程序逻辑控制器中保存对象名称”的选项）

19.3.1 上传一个站

用命令菜单PLC > Upload Station，可以从所选的可编程控制器中上传当前的组态和所有块到编程器上。

要作到这一步，在STEP 7当前项目中生成一个新站点，站的组态将存储在这个项目下。可以修改这个新站点的预定名称（例如，“SIMATIC 300-Station（1）”）。这个插入的站点在在线视图和离线视图中都会显示。

当打开一个项目时，这个命令菜单会被选中。在这个项目窗口选择一个对象还是在视图（在线或离线）中选择一个对象对这个菜单命令没有影响。

使用这个功能组态将变得更容易。

- 对于S7-300可编程控制器，被上传的实际硬件组态包括扩展机架，但是没有分布式I/O（DP）。
- 对S7-400可编程序控制器，只能上传的机架组态，没有扩展机架和分布式I/O。

对于没有分布式I/O的S7-300系统，要做的全部事情就是对模板作更详细的定义（订货号）并对它们进行参数赋值。

上传站点的限制

数据上传到编程器时应遵守下列限制：

- 块不包含任何参数、变量和标号的符号名。

- 块不包含任何注释。
- 包括所有的系统数据在内的完整的程序被上传，由此并非所有数据都能被进一步处理。
- 全局数据通信（GD）、组态符号相关的消息和组态网络的数据不能被进一步处理。
- 强制作业不能被上传到编程器上然后再下载到可编程序控制器上。

19.3.2 从S7 CPU中上传程序块

可以使用SIMATIC Manager，从CPU中上传S7块到编程器的硬盘上。在下列情况中，可以上传程序块到编程器上：

- 制作当前用户程序的备份。这个备份可以接着被重新下载，例如，维护人员维修后或CPU的存储器复位后。
- 从CPU中上传用户程序到编程器上并在编程器中编辑它，例如设备调试。这种情况下，不能访问到程序文档的符号或注释。因此，建议这一过程仅应用于维修目的。

19.3.3 在PG/PC中编辑上传的程序块

从CPU中上传块到编程器上的程序块有下列用途：

- 测试阶段中，可以在CPU上直接修改一个块，并为结果制作文档
- 可以通过装载功能从CPU的RAM装载存储器中上传当前内容到编程器上

注意

在线和离线工作时，时间标签冲突。

以下过程导可能致时间标签冲突，因此要避免。

当在线打开一个块时，产生时间标签冲突的原因：

- 在线所作的修改未保存到离线的S7用户程序中
- 离线所作的修改未下载到CPU

当离线打开一个块时，产生时间标签冲突的原因：

- 一个在线块的时间标签冲突被复制到离线的S7用户程序中，并且在离线时被打开

两种不同情况

当从CPU上传程序块到编程器时，记住有两种不同情况：

1. 块所属的用户程序在编程器上。
2. 块所属的用户程序不在编程器上。

这表示下面所列的程序部分不能下载到CPU中。这些内容是：

- 带有地址和注释的符号名的符号表
- 梯形逻辑或功能块图程序的段注释
- 语句表程序的行注释
- 用户定义的数据类型

19.3.3.1 PG/PC带有用户程序时编辑上传的程序块

按下列步骤编辑CPU中的块：

1. 在SIMATIC Manager中打开项目的在线窗口。
2. 选择在线窗口中“Blocks”文件夹。此时将显示已经上传的块。
3. 选择相应块，打开并对其编辑。
4. 选择菜单命令**File>Save**将所作的修改保存到编程器离线程序中。
5. 选择菜单命令**PLC<Download**向PLC下载所修改的块。

19.3.3.2 PG/PC不带有用户程序时编辑上传的程序块

按下列步骤编辑CPU中的块：

1. 在SIMATIC Manager中点击工具栏中“Accessible Nodes”按钮或选择菜单命令**PLC>Display Accessible Nodes**。
2. 从显示的列表中选择节点(“MPI=...”对象)并打开“Blocks”文件夹以显示块。
3. 打开程序块，如果需要还可以对其编辑、监视或复制。
4. 选择菜单命令**File>Save**并输入路径将其保存在编程器上。
5. 选择菜单命令**PLC<Download**向PLC下载所修改的块。

19.4 在可编程序控制器中删除程序块

19.4.1 清除装载/工作存储器并复位CPU

在下载用户程序到S7可编程序控制器之前，需要在CPU上完成一个存储器复位，以保证没有旧块保留在CPU上。

存储器复位的要求

CPU必须在STOP模式下才能完成存储器的复位（模式选择开关设置为STOP，或设成

RUN-P并用菜单命令PLC > Diagnostics/Settings > Operating Mode更改模式为STOP）。

在S7 CPU上执行存储器复位

当在S7 CPU上执行存储器复位时，可能出现下列情况：

- CPU被复位。
- 全部用户数据被删除（除了MPI参数以外的块和系统数据块（SDB）。
- CPU中断所有存在的连接。
- 如果数据保存在一个EPROM中（存储卡或集成EPPOM），随存储器复位后，EPROM的内容将复制到存储器的RAM区。

诊断缓冲区的内容和MPI参数将被保留。

在M7 CPU/FM上执行存储器的复位

当在M7CPU/FM上执行存储器复位时，可能出现下列情况：

- 初始状态被恢复
- 除了MPI参数以外的系统数据块（SDB）被删除
- CPU/FM断开所有存在的连接。将CPU从STOP切换到RUN后，用户程序被保留并继续运行

当出现严重错误时，使用“存储器复位”功能，通过删除工作存储器中当前系统数据块（SDB）并从只读存储器中重新装载SDB恢复M7CPU或FM的初始状态。在有些情况下，要求操作系统作暖起动。如需复位，可使用模式选择开关（切换到MRES位置）清除M7。只有当CPU/FM上使用的是RMOS32操作系统时才能够使用模式选择开关在SIMATIC M7CPU或FM上作复位。

19.4.2 删除可编程序控制器上的S7块

在CPU程序的测试阶段，删除CPU上的单个程序块可能是必要的。块被保存在CPU用户存储器EPROM中或RAM中（依据CPU和装载步骤）。

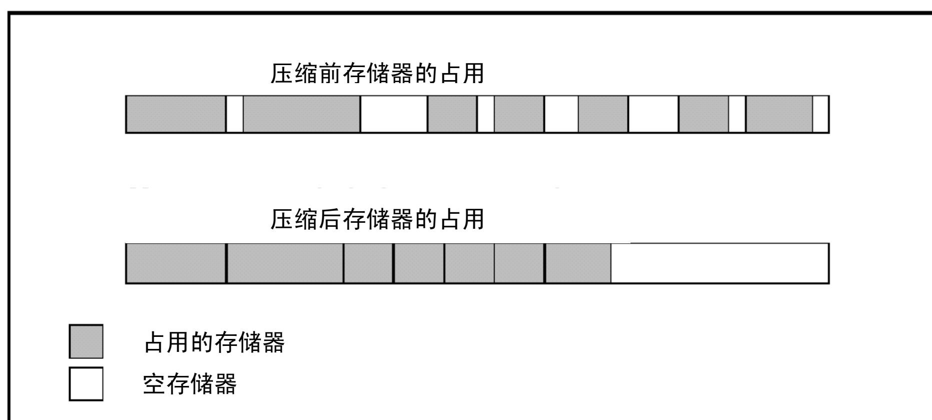
- RAM中的块可被直接删除。装载或工作存储器被占用的空间将空闲并可以重新使用。
- 随着CPU存储器的复位，集成EPROM中的块总是被复制到RAM区。RAM中的备份可以直接作删除，被删除的块会在EPROM中被标记为无效，直到下一次存储器复位或由于掉电而使没有备份的RAM数据丢失。在存储器复位或没有备份RAM电源掉电的情况下，“删除”的块从EPROM从中被复制到RAM中并且生效。使用RAM新的内容覆盖和删除集成EPROM中的程序块（例如，在CPU312中）。
- EPROM存储卡只能在编程器上被擦除

19.5 压缩用户存储器（RAM）

19.5.1 用户存储器里的间隙（RAM）

删除和重装块之后，用户存储器（装载和工作存储器）将出现间隙，减少了可用的存储区。使用压缩功能，使现有的块在用户存储器中无间隙地重新排列，产生一个连续的存储器空间。

下图显示了如何用压缩功能将占用存储器的块移到一起。



尽可能地在STOP模式下压缩存储器

只有在“STOP”模式下压缩存储器才能够将所有间隙闭合。在RUN-P模式时（模式选择开关设置），当前正在处理的块由于被打开了而无法移动。在RUN模式（模式选择开关设置）（写保护!）下压缩功能不起作用。

19.5.2 压缩S7 CPU存储器的内容

压缩存储器的方法

有下列两种方法可以压缩用户存储器：

- 如果存储器容量不够，载下载程序块时，将出现一个错误提示对话框。可以点击对话框中相应的按钮对存储器进行压缩。
- 作为预防措施，可以显示存储器可用空间(菜单命令PLC > Diagnostics/Setting > Module Information，“Memory”选择框)，如果需要，可以启动压缩功能。

步骤

1. 选择“Accessible Nodes”窗口中的S7程序或到项目在线视窗。
2. 选择菜单命令PLC > Diagnostics/Setting > Module Information。
3. 在出现的对话框中选择“Memory”选择框。如果CPU支持该功能，则在此页会出现一个压缩按钮。

20 使用变量监控表进行调试

20.1 使用变量表调试简介

使用变量表，可以保存各种测试环境。这样，在操作或进行维修和维护时，可以有效地进行测试和监控。项目中保存的变量表的数量没有限制。

当使用变量表进行测试时，有如下功能：

- **监视变量**
  该功能可以在编程器/PC机上显示用户程序中或CPU中每个变量的当前值。
- **修改变量**
  可以使用这个功能将固定值赋给用户程序或CPU中的每个变量。使用程序状态测试功能时也能立即进行一次数值修改。
- **使用外设输出与激活修改值**
  这两个功能允许CPU在停止模式下将固定值赋给每个I/O输出。
- **强制变量**
  可以使用这个功能给用户程序或CPU中每个变量赋予一个固定值，这个值是不能被用户程序覆盖的。

可以对以下变量进行赋值或显示数值：

- 输入、输出、位存储、定时器及计数器
- 数据块的内容
- I/O（外设）

在变量表中输入需要显示或修改的变量。

可以通过定义触发点和触发频率来决定变量什么时候、以什么样的频率被监视或赋予新值。

20.2 用变量表进行监视和修改的基本步骤

使用**监视**（Monitor）和**修改**（Modify）功能可按如下步骤进行：

1. 生成新的变量表或打开已存在的变量表。
2. 编辑或检查变量表的内容。
3. 用菜单命令**PLC > Connect to**，建立当前变量表与所需的CPU之间的在线连接。
4. 用菜单命令**Variable > Trigger**，选择合适的触发点并设置触发频率。
5. 菜单命令**Variable > Monitor**和**Variable > Modify**，切换监视和修改功能开与关。

6. 用菜单命令**Table > Save**或**Table > Save As**存储完整的变量表，以便在以后的任何时候再调用它。

20.3 编辑并存储变量表

20.3.1 生成并打开一个变量表

在监视或修改变量之前，必须生成一个变量表（VAT）并输入所需要的变量。生成变量表可选择以下方法之一：

在SIMATIC Manager中：

- 选择“Blocks”（块）文件夹，使用菜单命令**Insert > S7 Block > Variable Table**。 在对话框中，可以命名变量表（“Symbolic Name”文本框）。通过双击打开变量表。
- 选择一个连接或者一个在线视窗、可访问站点（accessible nodes）中的S7或M7程序。用菜单命令**PLC > Monitor/Modify Variables**，生成一个没有名称的变量表。

在“Monitor/Modify Variables（监视/修改变量）”中：

- 可以用菜单命令**Table > New**生成一个新的变量表，该表没有被赋给任何S7或M7程序。可以使用**Table > Open**打开已存在的表。
- 可以在工具栏中使用快捷键生成或打开变量表。

变量数一旦生成，可以存储、打印和多次使用该表进行监视和修改。

20.3.2 复制/移动变量表

你可以在一个S7/M7程序中的块文件夹中复制或移动变量表。

在复制或移动变量表时，应注意以下事项：

- 程序符号表中现有的符号将被刷新。
- 当移动变量表时，源程序符号表中的相应符号也将被移动到目标程序的符号表中。
- 从块文件夹中删除变量表时，S7/M7程序符号表中的相应符号也将被删除。
- 如果程序中包含一个具有相同名字的变量表，在复制变量表时，将被赋值更大的一个编号。
- 如果程序中包含一个具有相同名字的变量表，在复制时可以重新命名变量表（缺省为现有名字编号）。

20.3.3 存储变量表

存储变量表是为了再次测试程序时可以监视和修改变量。

1. 用菜单命令Table > Save存储变量表。
2. 如果已经生成变量表，必须命名变量表，例如“ProgramTest_1”。

存储变量表时，所有当前设定和表的格式都被存储，这意味着在菜单选项“Trigger”下的设置也被存储。

20.4 在变量表中输入变量

20.4.1 变量表中插入变量地址或符号

在变量表中输入需要修改或监视的变量。从“外部”开始，由“外”向“内”地输入原则，这意味着首先应该选择输入，然后是被输入影响的变量以及影响输出的变量，最后是输出。

例如，如果想监视输入位1.0，MW5以及输出字QB0，则在“Address”栏中输入以下内容：例如：

I 1.0
MW5
QB0

完整变量表的示例

下图所示是一个显示下述各栏的变量表：地址（Address）、符号（Symbol），显示格式（Monitor Format），监视值（Monitor Value）和修改值（Modify Value）

Var - @ Variable Table 1

Table Edit Insert PLC Variable View Options Window Help

@ "Variable Table 1" -- ZDt01_02_STEP7__AWL_1-10\SIMATIC 30...

	Address		Symbol	Display Format	Status Val	Force Valu
1	//OB1 Network 1					
2	I	0.1	"Pushbutton 1"	BOOL	true	
3	I	0.2	"Pushbutton 2"	BOOL	true	
4	Q	4.0	"Green light"	BOOL	false	
5	//OB1 Network 3					
6	I	0.5	"Automatic On"	BOOL	true	
7	I	0.6	"Manual On"	BOOL	true	
8	Q	4.2	"Automatic mode"	BOOL	true	true
9	//OB1 call FB1 for petrol engine on					
10	I	1.0	"PE_on"	BOOL	false	
11	I	1.1	"PE_off"	BOOL	false	
12	I	1.2	"PE_failur"	BOOL	false	
13	Q	5.1	"PE_preset_reached"	BOOL	false	
14	Q	5.0	"PE_on"	BOOL		true
15	//OB1 call FB1 for diesel engine on					
16	I	1.4	"DE_on"	BOOL	false	
17	I	1.5	"DE_off"	BOOL		

MPI = 3 (direct) Run

关于插入符号的注意

- 对需要修改的变量输入地址或符号，既可以在“符号”栏输入符号，也可以在“地址”栏输入地址。输入的条目自动写入正确的栏中。如果相应的符号已在符号表中定义，则符号栏或地址栏会自动填入。
- 只能输入已在符号表中定义过的符号。
- 符号必须与符号表中所定义的相同。
- 符号名中含有特殊字符则必须用引号括起来（例如，“Motor.off”，“Motor+Off”，“Motor-off”）。
- 要在符号表中定义新符号，可使用菜单命令**Options > Symbol Table**，也可以从符号表中复制符号然后粘贴到变量表中。

语法检查

在变量表中输入变量时，在每行的结束都会执行语法检查。任何不正确的输入都会被标为红色。如果将光标放在红色的行上，可以显示错误的原因。按F1可得到关于错误纠正的提示。

注意

如果用键盘而不用鼠标编辑变量表，应始能"Brief Information When Using the Keyboard"(使用键盘显示简短信息)特性。

如果需要，可以通过菜单命令Option>Customize修改变量表的设置，然后选择"General"标签栏。

最大限制

在变量表中每行最多可有255个字符。回车进入第二行是不可能的。一个变量表最多可有1024行，这是最大限制。

20.4.2 在变量表中插入一个连续的地址范围

1. 打开一个变量表。
2. 将光标指向地址列中的空闲位置，从该地址后将插入连续的地址范围。
3. 选择菜单命令Insert > Range of Variables。此时将出现"插入变量的范围"对话框。
4. 在"From Address"栏中输入起始地址。
5. 在"Number"栏中输入输入要插入列的数量。
6. 从显示清单中选择所需的显示格式。
7. 点击"OK"按钮。

此时在变量表中已经插入多个变量。

20.4.3 插入修改值

修改值作为注释

如果需要变量的"修改值"无效，可以使用菜单命令Variable > Modify Value as Comment。一个变量的修改值之前的注释符号"// "表示它已经无效。注释符号"// "插在"修改值"的前面，也可以代替调用菜单命令。通过再次调用菜单命令Variable > Modify Value as Comment 或删除注释符号，可以取消"修改值"的无效性。

20.4.4 定时器输入的上限

注意，下面是输入定时器的上限：

例如：W#16#3999（BCD格式的最大值）

举例

地址	监视格式	输入	修改值显示	解释
T 1	SIMATIC_TIME	137	S5TIME#130MS	转换为毫秒
MW4	SIMATIC_TIME	137	S5TIME#890MS	可用BCD格式表达
MW4	HEX	137	W#16#0089	可用BCD格式表达
MW6	HEX	157	W#16#009D	不能用BCD格式表达，所以监视格式不能选择为SIMATIC_TIME

注意

- 可以使用毫秒输入定时时间，但输入值会被作一些改变以适应时间格式。时间格式的大小要根据输入的时间值而定（137变成130ms，7ms被舍去）。
- 如果修改数据类型是WORD的变量，如IW1，被转换为BCD格式。但是，不是所有的位图都是有效的BCD码。如果将一个数据类型是WORD的变量表达为SIMATIC-TIME，则应用程序会自动转向缺省设置(这里是：HEX，请查看选择监视格式，缺省命令（“view”视窗菜单））， 这样数值就可以被显示了。

用于SIMATIC-TIME格式变量的BCD码

使用类型为SIMATIC_TIME的变量输入数值时要用BCD码。16个位具有以下含义：

| 0 0 x x | h h h h | t t t t | u u u u |

位15和14　总是零

位13和12　（标为xx）为位0至位11设置乘数：

　00=>乘数为10毫秒

　01=>乘数为100毫秒

　10=>乘数为1秒

　11=>乘数为10秒

位11至8为百位（hhhh）

位7至4为十位（tttt）

位3至0为个位（uuuu）

20.4.5 计数器输入的上限

注意下面计数器输入的上限：

计数器的上限是：C#999

W#16#0999（BCD格式的最大值）

例如：

地址	监视格式	输入	修改值显示	解释
C1	COUNTER	137	C#137	转换
MW4	COUNTER	137	C#89	可用BCD格式表达
MW4	HEX	137	W#16#0089	可用BCD格式表达
MW6	HEX	157	W#16#009D	不可以表达为BCD格式，不能选择COUNTER格式显示

注意

- 如果为计数器输入一个十进制数但却没有输入标识符C#，这个值会自动转为BCD格式（137变为C#137）。
- 如果修改数据类型是WORD的变量，如IW1，被转换为BCD格式。但是，不是所有的位图都是有效的BCD码。如果，数据类型为WORD的变量值无法表达为COUNTER，则应用程序会自动转向缺省格式（这里为：HEX，请查看选择监视格式，缺省命令（view 视窗菜单）），这样数值就可以被显示了。

20.4.6 插入注释行

注释行以标识符“// ”开始。

如果需要使一行或多行变量表无效（作为一个注释行），可以使用菜单命令**Edit > Row not Effect** 或工具栏中相应的符号 。

20.4.7 举例

20.4.7.1 在变量表中输入地址示例

允许的地址：	数据类型：	举例（英语助记符）：
Input | Output | Bit memory	BOOL	I 1.0 | Q 1.7 | M 10.1
Input | Output | Bit memory	BYTE	IB 1 | QB 10 | MB 100
Input | Output | Bit memory	WORD	IW 1 | QW 10 | MW 100
Input | Output | Bit memory	DWORD	ID 1 | QD 10 | MD 100
I/O （Input | Output）	BYTE	PIB 0 | PQB 1
I/O （Input | Output）	WORD	PIW 0 | PQW 1
I/O （Input | Output）	DWORD	PID 0 | PQD 1

允许的地址：	数据类型：	举例（英语助记符）：
Timers	TIMER	T 1
Counters	COUNTER	C 1
Data block	BOOL	DB1.DBX 1.0
Data block	BYTE	DB1.DBB 1
Data block	WORD	DB1.DBW 1
Data block	DWORD	DB1.DBD 1

注意：

“DB0...”不允许使用，因为它已被内部占用。

在强制数值窗口：

- 对S7-300模板进行强制时，只有输入、输出和I/O（输出）是可以的。
- 对S7-400模板作强制时，只有输入、输出、位存储和I/O（输入/输出）是可以的。

20.4.7.2 输入连续地址范围的示例

打开一个变量表并用菜单命令 **Insert > Range of Variables** 调用“Insert Range of Variables（插入变量范围）”对话框。

作为对话框的条目，下列各行标志位存储区被插入到变量表中：

- From address（开始地址）：M 3.0
- Number（数量）： 10
- Display format（显示格式）：BIN

地址	显示格式
M 3.0	BIN
M 3.1	BIN
M 3.2	BIN
M 3.3	BIN
M 3.4	BIN
M 3.5	BIN
M 3.6	BIN
M 3.7	BIN
M 4.0	BIN
M 4.1	BIN

注意示例中“地址”栏中从第八行以后字节地址改变。

20.4.7.3 输入修改值和强制值的示例

位地址

可能的位地址	允许修改/强制值
I1.0	true
M1.7	false
Q10.7	0
DB1.DBX1.1	1
I1.1	2#0
M1.6	2#1

字节地址

可能的字节地址	允许修改/强制值
IB 1	2#00110011
MB 12	B#16#1F
MB 14	1F
QB 10	'a'
DB1.DBB 1	10
PQB 2	-12

字地址

可能的字地址	允许修改/强制值
IW 1	2#0011001100110011
MW 12	w#16#ABCD
MW 14	ABCD
QW 10	B#(12,34)
DB1.DBW 1	'ab'
PQW 2	-12345
MW3	12345
MW5	s5t#12s340ms
MW7	0.3s or 0,3s
MW9	c#123
MW11	d#1990-12-31

双字地址

可能的双字地址	允许修改/强制值
ID 1	2#00110011001100110011001100110011
MD 0	23e4
MD 4	2
QD 10	dw#16#abcdef10
QD 12	ABCDEF10
DB1.DBD 1	b#(12,34,56,78)

可能的双字地址	允许修改/强制值
PQD 2	‘abcd’
MD 8	l#-12
MD 12	l#12
MD 16	-123456789
MD 20	123456789
MD 24	t#12s345ms
MD 28	tod#1:2:34.567
MD 32	p#e0.0

定时器

可能的“Timer”类型地址	允许修改/强制值	解释
T 1	0	转为毫秒
T 12	20	转为毫秒
T 14	12345	转为毫秒
T 16	s5t#12s340ms	
T 18	3	转为1s300ms
T 20	3s	转为1s300ms

修改定时器只影响其数值不影响状态。这意味着定时器T1的时间值可设为零，但却不会改变A　T1的逻辑操作结果。

输入字符5t、s5time，不区分大小写。

计数器

可能的“Counter”类型地址	允许修改/强制值
C 1	0
C 14	20
C 16	c#123

修改计数器只影响其数值不影响其状态。这意味着可将计数器C1的数值修改为零，却不会影响A　C1的逻辑操作结果。

20.5　建立与 CPU 的连接

为了监视或修改变量表（VAT）中输入的变量，必须与相应的CPU建立连接。可以将每个变量表与不同的CPU建立链接。

显示在线连接

如果在线连接存在，变量表窗口标题栏中的“ONLINE（在线）”项会显示该情况。状态栏将显示操作状态“RUN”、“STOP”、“DISCONNECTED”或“CONNECTED”，这取决于CPU。

建立与CPU在线连接

如果与所需要的CPU没有建立在线连接，使用菜单命令PLC > Connect To > …来定义与所需CPU的连接，以便进行变量的监视或修改。

中断与CPU的在线连接

使用菜单命令PLC > Disconnect，可以中断变量表和CPU的连接。

注意

如果用菜单命令Table > New，生成了一个未命名的变量表，它将自动连接到最后组态的CPU上（变量表中带有连接信息）。

20.6 监视变量

20.6.1 监视变量介绍

可用以下方法监视变量：

- 用菜单命令Variable > Monitor，激活监视功能。所选变量的数值依据所设定的触发点和触发频率显示在变量表中。如果设置触发频率为“Every Cycle（每一次扫描）”，可以用菜单命令Variable > Monitor，将监视功能切换成无效。
- 可以使用菜单命令Variable > Update Monitor Values，对所选变量的数值作一次立即刷新。所选变量的当前数值则显示在变量表中。

用ESC键放弃“监视”

如果在监视功能激活的状态下按ESC键，该功能则不经询问就退出。

20.6.2 设定变量表监控触发

可以在编程器上显示用户程序中每个变量在程序处理过程中的某一特定点（触发点）的当前数值，以便对它进行监视。

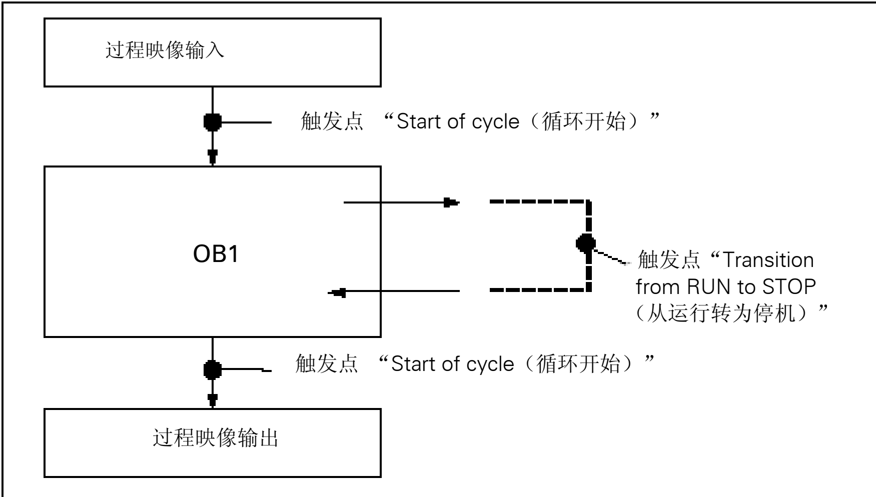
当选中一个触发点时，就决定了监视的变量在哪个时间点的数值被显示出来。

可以用菜单命令Variable > Trigger，设置触发点和触发频率。

触发	可能的设置
触发点	扫描开始 扫描结束 从RUN转换到STOP
触发频率	一次 每个扫描周期

触发点

下图所示为触发点的位置。



在“Status Value”栏中显示修改值，将监视的触发点设置在“Start of cycle”，将修改的触发点设置在“End of cycle”。

立即触发

可以用菜单命令Variable > Update Monitor Values，刷新所选变量的数值。这个命令即为“立即触发”，不参照用户程中的任何点，尽可能快地执行。这些功能主要用于CPU处于停机模式下的监视和修改。

触发频率

下表显示了触发频率对变量监视的影响：

	触发频率：一次	触发频率：每一个扫描
监视变量	依据触发点的设置，只触发一次	在定义的触发点上监视 当测试一个块时，可以准确地追踪处理的过程

20.7 修改变量

20.7.1 修该变量的介绍

可以利用下述方法进行变量修改：

- 用菜单命令Variable > Modify激活修改功能。在变量表中，根据触发点和触发频率的设定而对所选变量作的数值修改将应用到用户程序中。如果设置触发频率为“Every Cycle（每一扫描）”，可以用菜单命令Variable > Modify，将监视功能切换成无效。
- 用菜单命令Variable > Activate Modify Values，对所选变量的数据作一次立即刷新。

强制功能和使能外设输出（PQ）提供一些其它功能的可能性。

进行修改时应注意下面的情况：

- 只能对在变量表中可以看到的地址进行修改。
  如果缩小变量表的可视区域，有一些也许已修改了的值将不能看到。如果将变量表可视区域扩大，则可看到一些未修改的地址。
- 如果变量被修改则不能撤销（比如，用Edit > Undo）。



危险

在程序运行中修改变量值，如果功能或程序中出错可能导致对人身或财产的损害。在执行“修改”功能前，要确认不会有危险情况出现。

用ESC键放弃“修改”

如果在“Modifying（修改）”功能进行过程中使用ESC键，该功能无询问退出。

20.7.2 设置变量表触发功能

可以在程序运行中的某一个特定点（触发点）给用户程序中的变量赋予固定值（一次或每一次扫描）。

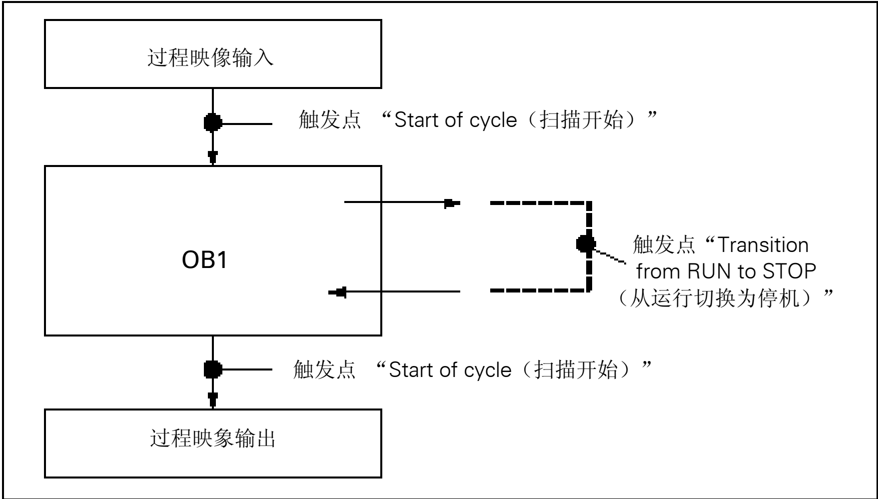
当选中一个触发点时，就决定了监视的变量在哪个时间点的数值被显示出来。

可以用菜单命令Variable > Trigger，设置触发点和触发频率。

触发	可能的设置
触发点	扫描开始 扫描结束 从运行转为停机
触发频率	一次 每一次扫描

触发点

下图所示为触发点的位置。



所示为触发点的位置：

- 修改输入只适用于触发点“Start of cycle”（对应于用户程序OB1的开始），因为在修改后将更新过程映象输入区。
- 修改输出只适用于触发点“End of cycle”（对应于用户程序OB1的结束），否则用户程序将覆盖过程映像输出区。

在“Status Value”栏中显示修改值，将监视的触发点设置在“Start of cycle”，将修改的触发点设置在“End of cycle”。

当变量修改时，下列情况适用不同的触发点：

- 如果触发频率设置为“Once”，而所选变量不能被修改时，会显示信息。
- 如果触发频率设置为“Every cycle”则无信息出现。

立即触发

可以使用菜单命令Variable > Activate Modify Values，对所选变量的数值进行修改。这个命令即为“立即触发”，不参照用户程中的任何点，尽可能快地执行。这个功能主要用于在CPU停机模式下修改。

触发频率

下表所示为触发频率的设定对变量修改的影响：

	触发频率：一次	触发频率：每一次扫描周期
修改变量	激活一次，与触发点无关，可以将数值赋给变量一次	在定义的触发点进行修改 通过赋予固定数值，可以为用户程序模拟某一情形，用这个功能可调已编好的功能

20.8 强制变量

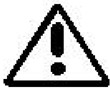
20.8.1 强制变量时的安全措施



注意人员伤害和财产损失

注意，当使用“强制”功能时，任何不正确的操作都可能会：

- 危及人员的生命或健康，或者
- 造成设备或整个工厂的损失





小心

- 在开始强制功能之前必须检查确保同一时间在同一CPU上没有其他人在执行该功能。一个强制作业只能用菜单命令Variable > Stop Forcing来删除或终止。关闭强制数值窗口或退出 “Monitoring and Modifying Variables”应用程序并不能删除强制作业。
- 强制功能不能被取消（例如，用Edit > Undo）。
- 请阅读有关信息，了解强制和修改之间的差别。
- 如果一个CPU不支持强制功能，则在变量菜单中与强制有关的所有菜单命令都不能激活。

如果用菜单命令Variable > Enable Peripheral Output，使输出禁用失效，所有被强制的输出模板输出它们的强制值。

20.8.2 强制变量的介绍

可以给用户程序的每个变量赋予固定值，即使CPU中正在执行的用户程序也不能够改变或覆盖变量赋予的值。实现这一功能的要求是CPU支技该功能（如，S7-400的CPU）。通过将固定值赋给变量这一功能，可以为用户程序设置特定的情形并用该方法对已编程的功能进行测试。

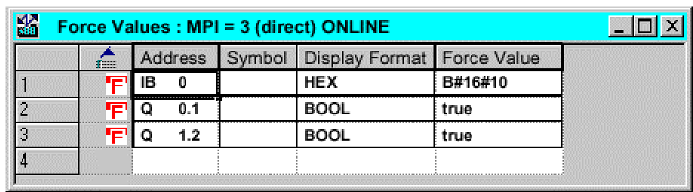
“强制数值（ForceValues）”窗口

只有当“强制数值”窗口处于激活状态，才能选择用于强制的菜单命令。

要显示这个窗口可选择菜单命令Variable > Display Force Values。

在CPU中打开一个“强制数值”窗口，被强制的变量和它们各自的强制值一起显示在窗口中。

强制数值窗口示例



当前在线连接的名字显示在**标题栏**中。

状态栏中显示的是从CPU中读出的强制作业的日期和时间。

如果没有激活的强制作业，该窗口是空的。

在“强制数值”窗口中**变量**不同的**显示**方法有以下含义：

显示	含　义
黑体	该变量在CPU中已被赋予固定值
正常	该变量正在被编辑
灰色	模板的变量在机架上不存在/未插入或者变量地址错误，显示错误信息

使用变量表中可强制的地址

如果在强制变量窗口中从变量表中输入一个变量，选择变量表及所需的变量，然后调用菜单命令Variable > Force values打开强制值窗口。将可以强制的变量输入到强制值窗口中。

使用CPU中的强制作业或建立新的强制作业

如果“强制数值”窗口是打开并且激活，则有另外的信息显示：

- 如果确认，该窗口中的改变将会被CPU中已存在的强制作业覆盖。用菜单命令Edit > Undo，可以重新存储前一窗口的内容。
- 如果取消，当前窗口中的内容就保留下来。
  然后可以用菜单命令Table > Save As，将“强制数值”窗口的内容存为一个变量表，或者选择菜单命令Variable > Force：这样就将当前窗口的内容写到CPU中作为一个新的强制作业。

变量的监视和修改只能在变量表中进行，不能在“强制数值”窗口。

删除强制数值

调用菜单命令Variable > Display Force Values打开强制数值窗口。然后调用菜单命令Variable > Delete Force从所选的CPU中删除强制值。

存储强制数值窗口

可以将强制数值窗口中的内容存到一个变量表中。使用菜单命令Insert > Variable Table，可以在一个强制数值窗口中重新插入已存储的内容。

在强制数值窗口中关于符号的提示

除非从其它的没有符号的应用程序中打开“监视和修改变量”应用程序，否则最后激活的窗口中的符号会被输入。

如果无法输入符号名，则“Symbol（符号）”这一栏被隐藏了。这种情况下，菜单命令 **Option > Symlol Table**没有激活。

20.8.3 强制和修改变量的区别

下表总结了强制和修改的区别：

特点/功能	S7-400(包括CPU 318-2DP)强制功能	S7-300(不包括CPU 318-2DP)强制功能	修改
位存储(M)	Yes	-	Yes
定时器和计数器(T、C)	-	-	Yes
数据块(DB)	-	-	Yes
外设输入(PIB、PIW、PID)	Yes	-	-
外设输出(PQB、PQW、PQD)	Yes	-	Yes
输入和输出(I、Q)	Yes	Yes	Yes
用户程序可覆盖修改/强制值	-	Yes	Yes
无中断替换强制数值	Yes	Yes	-
当应用程序退出时变量仍保持其数值	Yes	Yes	-
与CPU的连接断开后变量仍保持其数据	Yes	Yes	-
允许的寻址错误： 如：IW1 修改/强制值：1 IW1 修改/强制值：0	-	-	最后修改的成为效值
设置触发	总是立即触发	总是立即触发	一次或每次扫描周期
功能只影响激活窗口中可视区域的变量	影响所有强制值	影响所有强制值	Yes

注意

- “使能外设输出”时，在相应输出模板上外设输出的强制数值有效；而外设输出的修改值却无效。
- 使用强制功能，变量总是带有强制的数值。这些数值可以在用户程序的读取访问过程中被读取，对数值所有形式的写访问都无效。
- 使用永久修改功能时，对程序的读访问是有效的并可保持到下一触发点。

21 测试程序状态

通过显示程序状态（RLO，状态位）或为每条指令显示相应寄存器内容的方法对程序进行测试。可以在“Customize”对话框中定义在“LAD/FBD”画面中的显示范围。在“LAD/STL/FBD：Programming Blocks”窗口中，使用菜单命令**Options > Customize**打开这个对话框。



警告

在过程运行时测试程序，如功能或程序出错，会对人员或财产造成严重损害。

在开始测试功能之前，确认不会有危险情况出现。

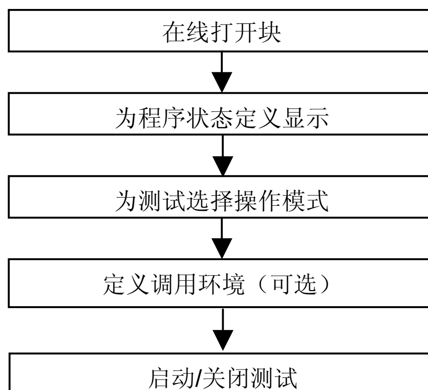
要求

显示程序状态，必须满足下列要求：

- 存储没有错误的程序，并且下载到CPU。
- CPU运行并且用户程序执行。

监视程序状态的基本步骤

建议不要调用整个程序进行调试，而应一个块一个块地调用并单独调试。应该从调用分层嵌套中最外层的块开始，例如，在OB1中调用它们，通过监视和修改变量功能生成测试的环境。



设置断点，并在单步模式下执行程序，必须设置为测试操作模式（见菜单命令**Debug > Operation**）。这些测试功能在过程操作模式下是不可能的。

21.1　程序状态显示

程序状态的显示是循环刷新的。它从所选择的程序段开始。

在LAD和FBD编辑器中的预设颜色代码

- 状态满足：绿色实线
- 状态不满足：兰色点线
- 状态未知：黑色实线

在“LAD/FBD”标签中，使用菜单命令**Options > Customize**，可改变线型及颜色的预置。

元素的状态

- 触点的状态是：
  - 如果该地址有“1”值则满足。
  - 如果该地址有“0”值则不满足。
  - 如果该地址的值不知道则为未知。
- 带有使能输出（ENO）的元素状态对应于以ENO输出值为地址的触点的状态。
- 带有Q输出的元素状态对应于该地址值的触点状态。
- 如果BR位在调用功能后被置位，则调用（CALL）的状态满足。
- 如果跳转被执行则跳转指令的状态满足，即意味着跳转条件满足。
- 带有使能输出（ENO）的元素，如果使能输出未被连接则该元素显示为黑色。

线的状态

- 线的状态如果未知或没有完全运行则是黑色的。
- 在能流开始处线的状态总是满足的（“1”）。
- 并行分支开始处线的状态总是满足的（“1”）。
- 如果一个元素和它前面的线的状态都满足则该元素后面的线的状态满足。
- 如果NOT指令前面的线的状态不满足（相反）则NOT指令后面的线状态满足。
- 在下列情况下，许多线交出点后面的线状态可以满足：
  - 与**之前**至少有一条线的状态被满足。
  - 分支前的线的状态满足。

参数状态

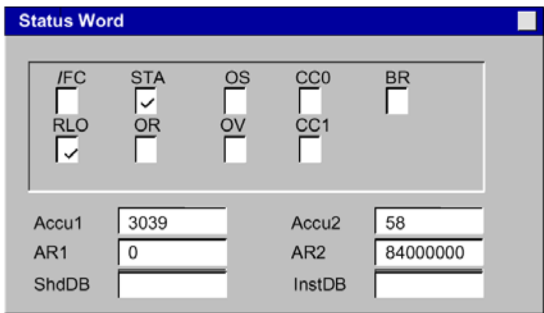
- **黑体类型**的参数值是当前值。
- 细体类型的参数值来自前一个扫描；该程序区在当前扫描循环中未被处理。

21.2 在单步模式/断点下进行测试应了解的信息

在单步模式下进行测试，可以做以下事情：

- 一条指令一条指令地执行程序（在单步模式下）
- 设置断点

“在单步模式下测试”的功能并不是在所有的可编程控制器上都能实现（参考相关的可编程控制器的资料）。



要求

- 必须对测试模式进行设置。在过程操作模式下单步测试模式是不可能的（见菜单命令 Debug > Operation）。
- 只有使用STL编程时，才可以使用单步执行模式测试。使用LAD或FBD生成的块，必须用菜单命令View > STL改变为STL编程语言。
- 该块不能被保护。
- 块必须在线打开。
- 已打开的块不能在编辑器中进行修改。

断点的数量

断点的数量依据下列情况而变化：

- 已设置的断点数量
- 变量状态运行的数目
- 程序状态运行的数目

参考可编程控制器相关资料，查看是否支持单步执行模式。

在“Debug（调试）”菜单中，可以找到菜单命令用来设置、激活或删除断点，也可以用断点栏中的快捷键选择这些菜单命令，使用菜单命令View > Breakpoint Bar可以显示断点栏。

允许的测试功能

- 监视/修改变量

- 模板信息
- 操作模式

危险（Danger）

在HOLD（保持）模式下注意整个系统风险。

21.3　在 HOLD（保持）模式应该了解的信息

如果程序遇到断点，可编程控制器进入HOLD操作模式。

HOLD模式下LED的显示

- LED　RUN　 闪烁
- LED　STOP　亮

在HOLD模式下程序的处理

- 在HOLD模式下不处理S7指令码，这意味着没有优先级被进一步处理。
- 所有定时器被冻结：
  - 不处理任何定时器单元
  - 所有监控时间停止
  - 时间控制的基本时钟被停止
- 实时时钟继续运行
- 为安全原因，在HOLD模式下输出总是禁止的（“输出禁止”）。

HOLD模式下电源掉电后的性能

- 装有后备电池的可编程控制器在HOLD模式下，在电源故障并恢复时将进入STOP模式并保持在STOP模式。CPU不执行自动重启动(暖启动)。在STOP模式下可以决定过程如何继续（例如，设置/清除断点，执行一个手动再启动）。
- 没有后备电池的可编程控制器不具有“记忆”功能，所以当电源恢复后执行一个自动暖启动，而不考虑以前的操作模式。

21.4　数据块的状态

版本5以上的STEP 7都可以在数据视窗中在线监控数据块。监控功能可以由在线数据块也可以由离线数据块激活。在这两种情况下，都可以显示可编程控制器中数据块内容。

在启动程序状态之前，不能修改数据块。如果在线数据和离线数据块之间有结构差异（声明），可以根据需要将离线数据块直接下载到可编程控制器中。

数据块必须位于“data view（数据视窗）”中，以使在线数值可以显示在“Actual Value（实际数值）”栏中。只有显示窗口中可以看到的数据块部分才能刷新。在状态激活时，不能切换为声明视窗。

在更新过程中，在状态栏中将显示一个绿框，并显示操作模式。

数值分别以各自的数据类型显示；其格式不能修改。

在结束程序状态后，“Actual Value（实际数值）”栏将显示程序状态之前的有效内容。不能将刷新的在线数值上传至离线数据块。

更新数据类型：

在共享DB和实例数据块的所有声明（in/out/in-out/stat）中，可以更新所有基本数据类型。

有些数据类型不能更新。当程序状态激活时，在“Actual Value（实际数值）”栏中包含没有更新的数据区域将显示为灰色背景。

- 复合数据类型DATE_AND_TIME和STRING不能更新。
- 在复合数据类型ARRAY、STRUCT、UDT、FB和SFB中，只有那些基本数据类型元素才能被更新。
- 在一个背景数据块的INOUT声明中，只显示复合数据类型指针，不显示数据类型的元素本身。指针不能更新。
- 参数类型不能更新。

21.5　为程序状态设置显示

可以设置程序状态的显示语言为STL、FBD或LAD。

按如下步骤进行显示设置：

1. 选择菜单命令**Options > Customize**。
2. 在“Customize”对话框中，选择STL或LAD/FBD。
3. 为程序测试选择所需的选项，在状态域中可显示下面对象：

激　活	显　示
状态位	状态位；状态字的第2位
RLO	状态字的第1位；显示逻辑运算或算术比较的结果
标准状态	累加器1的内容
地址寄存器1/2	各自带有间接寻址的指针地址
Accu2	累加器2的内容
DB寄存器1/2	数据块寄存器的内容，第1个和/或第2个打开的数据块的内容

激　活	显　示
间接	间接存储器参考；指针参考(地址)，无地址内容参考；只能对存储器间接寻址，而不能对寄存器间接寻址。 如果在语句中有相应的指令，则显示定时器字或计数器字的内容。
状态字	状态字的所有的状态位

21.6　设置测试模式

步骤：

1. 使用菜单名令Debug > Operation显示设置的测试环境。
2. 选择所需的运行模式。可以选择测试运行模式或处理运行模式。

运行模式	说　明
测试运行	可以不受限制地使用所有测试功能。 会明显增加CPU扫描循环时间。
处理运行	测试功能程序状态是受限制的，以保证测试占用扫描循环最小的负荷。 • 不允许条件调用 • 编写的循环状态显示在返回点被禁止 • 不允许HOLD测试功能和单步执行

注意

当通过向CPU赋值参数来设置运行模式时，只能通过更改参数来改变运行模式，另外也可以在显示的对话框中改变运行模式。

22 使用模拟软件（可选软件包）进行测试

22.1 使用模拟程序 S7 PLCSIM（可选软件包）进行测试

可以使用可选软件包PLC Simulation（模拟PLC），在计算机或编程器（如PG740）上模拟可编程控制器的运行并测试程序。由于该模拟功能可完全由STEP 7软件实现，不需要任何S7硬件（CPU或信号模板）。使用模拟的S7 CPU，可以对S7-300和S7-400 CPU作程序测试和故障诊断。

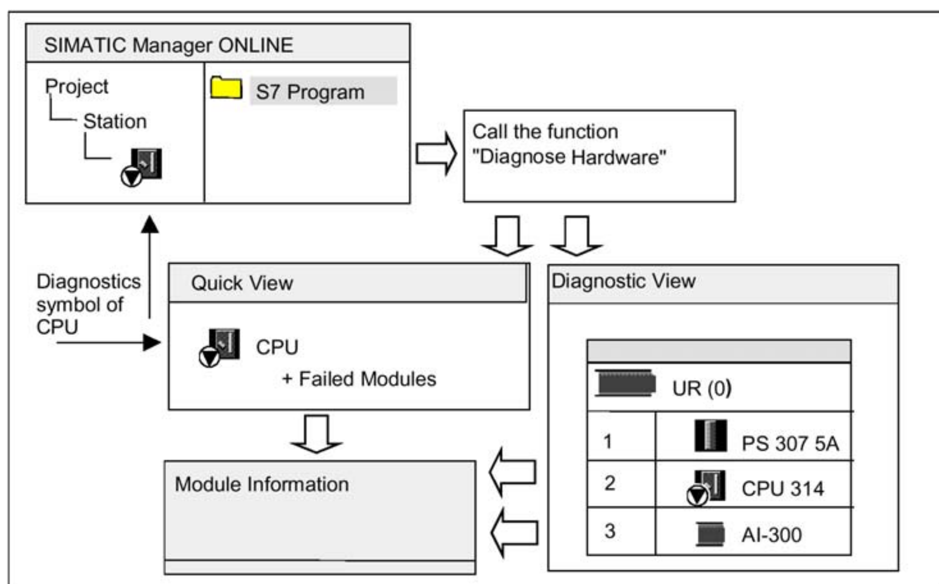
模拟软件为监视和修改各种用户程序中使用的参数提供了一个简单的用户界面（如，切换输入开关接通和断开）。程序由模拟的CPU处理时，还可以在STEP 7中使用各种应用程序。例如，可以用变量表监视和修改变量。

23 诊断

23.1 硬件诊断和故障排除

可以通过观察诊断符号来判断一个模板是否有诊断信息。诊断符号可显示相应模板的状态，对于CPU，还可显示操作模式。

诊断符号可显示在项目在线窗口中，以及当调用“Diagnose Hardware（诊断硬件）”功能时显示在快速视窗（缺省设置）或诊断视窗中。更详细的诊断信息显示在“Module Information（模板信息）”应用程序中，可以通过双击快速视窗或诊断视窗中的诊断符号起动该应用程序。



如何进行故障定位

1. 用菜单命令View > Online，打开项目的在线窗口。
2. 打开所有的站，以便看到其中组态的可编程模板。
3. 查看哪个CPU显示指示错误或故障诊断符号。可以使用F1键调用对诊断符号进行解释的在线帮助。
4. 选择要检查的站。
5. 选择菜单命令PLC > Diagnostics/Settings > Module Information，显示该站中CPU的模板信息。
6. 选择菜单命令PLC > Diagnostics/Settings > Diagnose Hardware，显示CPU的“quick view”（快速视窗）”及本站中有故障的模板。快速视窗的显示被设作缺省

设置（菜单命令Option > Customize，“View”标签）。

7. 在快速视窗中选择故障模板。
8. 点击“Module Information（模板信息）”按钮，以获得该模板的诊断信息。
9. 点击快速视窗中的“Open Station Online”按钮可显示诊断视窗。诊断视窗中包含了该站中按插槽顺序排列的所有模板。
10. 双击诊断视窗中的模板以显示其模板信息。用这种方式，还可以得到那些没有故障因而没有显示在快速视窗中的模板信息。

没有必要执行上述全部的步骤，当得到所需的诊断信息后就可以停止诊断工作。

23.2 在线视窗中的诊断符号

诊断符号显示在项目在线窗口和硬件组态界面中在线窗口。

诊断符号便于检查故障。只需看一眼模块符号，就知道有没有诊断信息。如果没有出现故障，则在模板类型上不显示诊断符号。

如果模板有诊断信息，则会在模板的符号上增加一个诊断符号，或显示的模板符号对比度降低。

模板的诊断符号(例：FM / CPU)

符　号	模　式
	预置组态与实际组态不匹配：被组态的模板不存在，或者插入了不同类型的模板
	故障：模板出现故障 可能的原因：诊断中断，I/O访问错误，或检查到故障LED
	不能进行诊断。原因：无在线连接或该模块不支持模板诊断功能（如：电源或子模板）

操作模式的诊断符号(例如：CPU)

符　号	模　式
	起动（STARTUP）
	停机（STOP）
	停机（STOP） 在多CPU操作模式下由另一个CPU触发的停机

符　号	模　式
	运行（RUN）
	保持（HOLD）

强制的诊断符号

符　号	含　义
	在该模板上有变量被强制，即在模板的用户程序中有变量被赋予一个固定值，该数据值不能被程序改变。 强制符号还可以与其它符号组合在一起显示（这里是与运行（RUN）模式符号一起显示）。

更新诊断符号的显示

必须激活相应的窗口。

- 按F5，或
- 在窗口中选择菜单命令**View > Update**。

23.3　硬件诊断：快速视窗

23.3.1　访问快速视窗功能

快速视窗可以快速进行“Diagnosing Hardware（硬件诊断）”，但其显示的信息没有硬件组态中诊断视窗所显示的信息详细。当调用“Diagnos Hardware”功能时，快速视窗作为缺省设置显示。

显示快速视窗

在SIMATIC Manager中使用菜单命令**PLC > Diagnostics/Settings > Diagnose Hardware**调用该功能。

可以在下列情况下使用菜单命令：

- 如果在一个在线项目窗口中选中了一个模板或一个S7/M7程序。
- 如果在“Accessible Nodes（可访问站点）”窗口中选中了一个站（“MPI=…”）并且该选项是一个CPU。

从显示的组态列表中，可以选择需要显示其模板信息的模板。

23.3.2 快速视窗中的信息功能

下列信息会显示在快速视窗中：

- 在线连接到CPU的数据
- CPU的诊断符号
- 被CPU检测出故障的诊断符号（例如，诊断中断、I/O访问错误）
- 模板类型和地址（机架、槽、带有站号的DP主站系统）

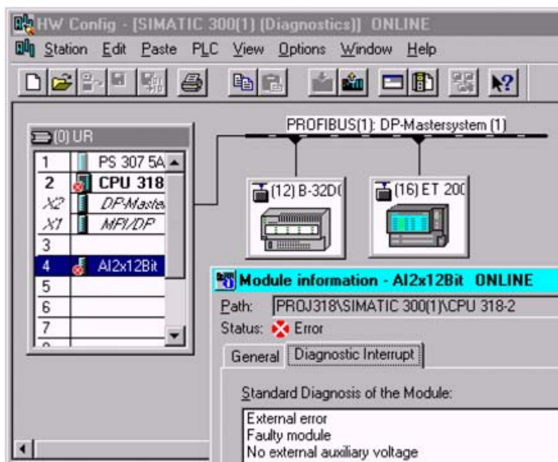
快速视窗中的其它诊断选项

- **显示模板**信息
  通过点击“Module Information（模板信息）”按钮调用该对话框。对话框依据所选模板的诊断能力显示详细的诊断信息。可以通过CPU的诊断信息显示诊断缓冲区中的各个条目。
- **显示诊断视窗**
  使用“Open Station Online（打开在线站点）”按钮可以打开一个对话框，与快速视窗相比，该对话框中包含整个站的图形概述以及组态信息。它主要显示“CPU/Faulty Modules（CPU/故障模板）”列表中被高亮选中的模板信息。

23.4 硬件诊断：诊断视窗

23.4.1 调用诊断视窗

使用这种方法可以为机架上的所有模板打开“Module Information”对话框。诊断视窗（组态列表）显示在机架层中站的实际结构，以及DP站及其模板。



注意

- 如果离线的组态表已经打开，可以使用菜单命令**Station > Open Online**，打开组态列表的在线视窗。
- 根据模板的诊断能力，在“Module Information”对话框中可以显示不同的信息标签。
- 在“Accessible Nodes”窗口中，只能显示有站地址（MPI、PROFIBUS、以太网地址）的模板。

从SIMATIC Manager的在线项目视窗中调用

1. 在SIMATIC Manager的项目视窗中，使用菜单命令**View > Online**建立与PLC的在线连接。
2. 选择一个站并双击打开。
3. 然后打开其中的“Hardware”对象。打开诊断视窗。

可以选择一个模板，使用菜单命令**PLC > Diagnostics/Settings > Module Information**调用其模板信息。

从SIMATIC Manager的离线项目视窗中调用

执行以下步骤：

1. 在SIMATIC Manager的项目视窗中选择一个站并双击打开。
2. 然后打开其中的“Hardware”对象，打开组态列表。
3. 选择菜单命令**Station > Open Online**。
4. 模板（如，CPU）所决定的站点组态及硬件组态的诊断视窗就打开了。模板的状态由符号来指示。各种符号的含义可参考在线帮助。故障模板以及已经组态但找不到的模板分别列在不同的对话框中。在这个对话框中可以直接去找选中的模板（“Go To”键）。
5. 双击模板的符号，带有标签（依模板类型而不同）的对话框可以给出有关模板状态的详细分析。

从SIMATIC Manager的“Accessible Nodes”窗口中调用

执行以下步骤：

1. 使用菜单命令**PLC > Display Accessible Nodes**，在SIMATIC Manager中打开“Accessible Nodes”窗口。
2. 在“Accessible Node”窗口选择一个站。
3. 选择菜单命令**PLC > Diagnostics/Settings > Diagnose Hardware**。

注意

在“Accessible Nodes”窗口中，只能显示有站地址（MPI、PROFIBUS、以太网地址）的模板。

23.4.2 诊断视窗中的信息功能

与快速视窗相比，诊断视窗显示已在线连接的整个站的组态信息。包括：

- 机架配置
- **所有**组态模板的诊断符号，从这里可以读出每个模板的状态以及CPU模板的操作模式。
- 模板类型、订货号和地址细节以及组态注释。

诊断视窗中其他的诊断选项

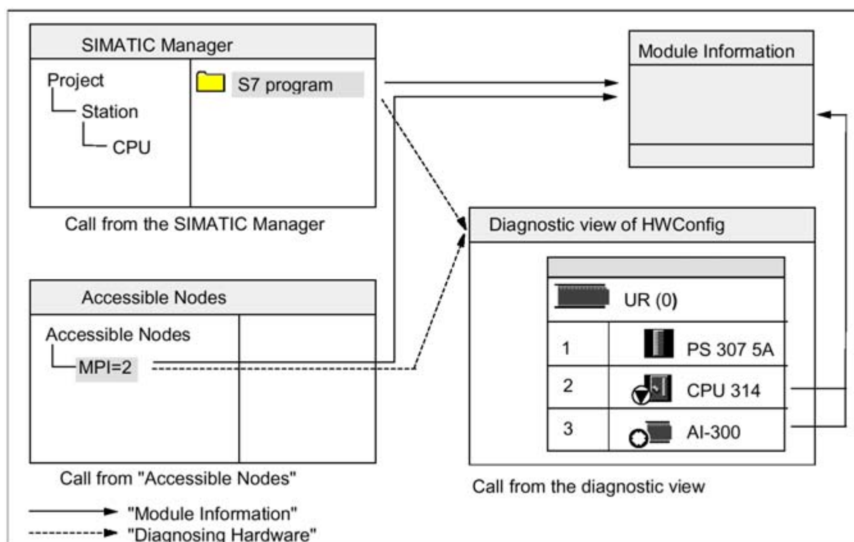
双击一个模板，可以显示该模板的操作模式。

23.5 模板信息

23.5.1 显示模板信息的选项

可以从不同的开始点显示“Module Information（模板信息）”对话框。下列步骤是调用模块信息经常使用的方法：

- 在SIMATIC Manager中从项目的“在线”或“离线”视窗调用。
- 在SIMATIC Manager的“Accessible Nodes（可访问节点）”窗口中调用。
- 在硬件组态的诊断视窗中调用。



为了要显示**模板及其站地址的**状态信息，需要对可编程控制器进行在线连接。可通过项目的在线视窗或通过“可访问站点”窗口建立连接。

23.5.2 模板信息功能

在“Module Information”对话框中，不同的标签对应不同的模板信息功能。在激活状态下显示时，只显示那些与所选模板相关的标签。

功能/标签	信　息	用　途
概述	所选模板的标识数据；比如，订货号、版本号、状态、机架中的插槽	可将所插模板的在线信息与所组态模板的数据进行比较
诊断缓冲区	诊断缓冲区中事件总览以及所选中事件的详细信息	查找引起CPU进入STOP模式的原因，并在选中的模板上评估导致绍停机原因的事件 使用诊断缓冲区还可在以后对系统错误进行分析，查找停机原因并对出现的每个诊断事件进行追踪和分类
诊断中断	所选模板的诊断数据	评估模板故障的原因
DP从站诊断	所选DP从站（参照EN50170）的诊断数据	评估DP从站中的故障原因
存储器	存储能力。所选的CPU或M7功能模板的工作存储器和装载存储器以及具有保持功能的存储器的当前使用情况	在新块或扩展块传送到CPU之前，可检查CPU/功能模板的装载存储器中是否有足够的空间，或者需要压缩存储器内容
扫描循环时间	所选CPU或M7功能模板最长、最短和最近一次的循环扫描时间	用于检查所组态的最小循环时间、最大循环时间和当前循环时间
时间系统	当前时间、操作小时数以及同步时钟的信息（同步周期）	可显示并设置模板的时间和日期，检查时间同步
性能数据	所选模板（CPU/FM）的地址区和可使用的块	在生成用户程序之前或之中检查CPU是否满足执行用户程序的要求。例如，装载存储区的大小或过程映像的大小
块(可在“Performance Data”标签中打开)	显示所选模板所有可用的块类型，包括OB、SFB、SFC列表，可将这些块用于该模板	检查用户程序中可包含或调用哪些可在所选CPU上运行的标准块
通讯	传送速率，通讯连接概述，通讯负载以及所选模板在通讯总线上最大的信息容量	用来决定CPU或M7 FM的连接数量和种类，以及正在使用的数量
堆栈	**堆栈**标签：只能在STOP模式或HOLD模式下调用 显示所选模板的B（块）堆栈。然后还可以显示I（中断）堆栈、L（局域）堆栈以及嵌套堆栈。可以跳转到中断块的故障点。	可判明引起停机的原因并对块进行修改

显示附加信息

对每个标签可显示以下信息：

- 到所选模板的在线路经。
- 相应CPU的操作模式（例如：运行、停机）。
- 所选模板的状态（例如：出错、OK）。

- 所选模板的操作模式（例如：运行、停机），如果模板具有操作模式（如CP342-5）。

如果从“Accessible Nodes”窗口中打开非CPU模板的模板信息中，则不能显示CPU本身的操作模式和所选模板的状态。

同时显示多个模板

可以同时显示多个模板的模板信息。要做到这一点，必须改变各个模板的背景，选择另一个模板，然后调用模板信息，“Module Information（模板信息）”对话框就会显示出来。每个模板只能打开一个对话框。

刷新模板的信息显示

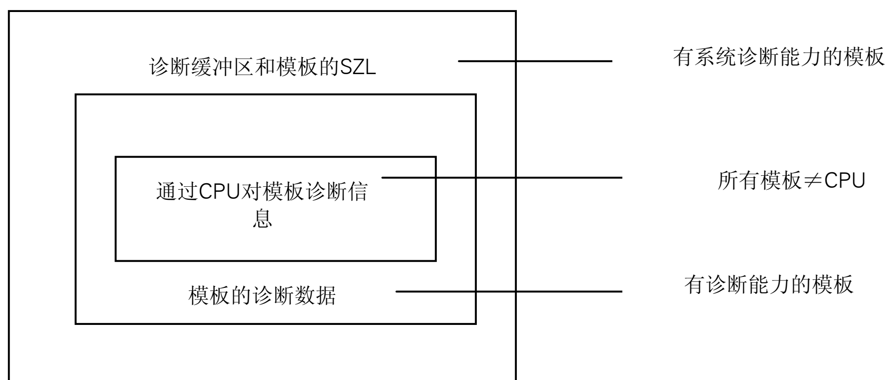
在“Module Information”对话框中，每次切换标签时，都可以重新读取模板数据。但是，当一页显示之后，其内容不再刷新。如果点击“Updete（刷新）”按钮，可以在不改变标签的情况下从模板中读取数据。

23.5.3 模板类型所决定的模板信息范围

可以评估和显示的信息范围取决于：

- 所选模板，以及
- 从哪个视窗调用模板信息
  当从组态列表的在线视窗或项目窗口调用时可得到全范围的信息。
  从“Accessible Nodes（可访问站点）”窗口调用只能得到有限范围内的信息。

根据信息的范围，模板可分为几类：“有系统诊断能力”、“有诊断能力”或“无诊断能力”。下图所示这些类别：



- 有系统诊断能力的模板，如模板FM 351和FM 354
- 模拟信号模板大多具有诊断能力
- 数字信号模板大多没有诊断能力

显示的标签

下表显示了“模板信息”对话框中每种类型的模板都显示哪些特性标签。

标签	CPU或 M7 FM	有系统诊断能力的模板	有诊断能力的模板	无诊断能力的模板	DP从站
常规	有	有	有	有	有
诊断缓冲区	有	有	---	---	---
诊断中断	---	有	有	---	有
存储器	有	---	---	---	---
扫描循环时间	有	---	---	---	---
系统时间	有	---	---	---	---
性能数据	有	---	---	---	---
堆栈	有	---	---	---	---
通讯	有	---	---	---	---
DP从站诊断	---	---	---	---	有
H状态[1)]	有	---	---	---	---
1） 只有H系统的CPU					

除了标签特性页的信息之外，有操作模式的模板还会显示操作模式。当从在线的组态列表中打开对话框时，会从CPU角度来显示模板的状态（例如，OK、故障、模板不存在）。

23.5.4 显示Y-Link后面链接的PA现场设备和DP从站的模板状态

对于STEP 7 V5.1 SP3，在DP/PA链接后(IM 157)可以评估DP从站和PA现场设备的模板状态。

它将影响下面的组态：

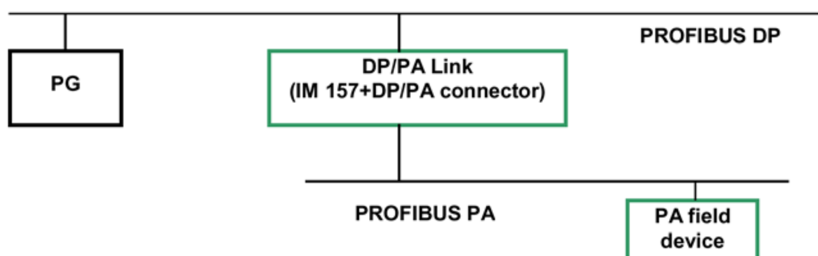
- 带DP/PA连接器的IM 157，用于连接PROFIBUS-PA
- IM 157作为一个冗余的接口模板，用于连接非冗余PROFIBUS- DP(“Y-link”)

在该组态中，编程器通过DP/PA链接连接到相同的PROFIBUS子网。

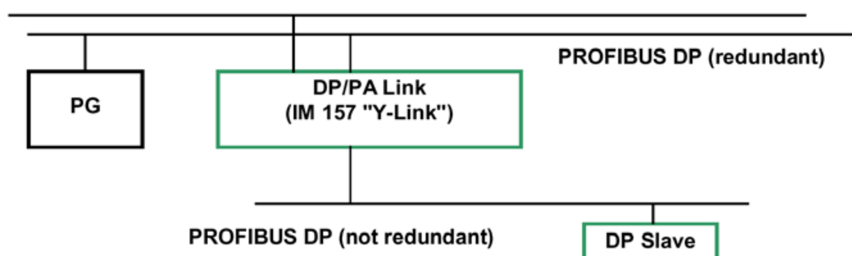
此外，还有一个选项，通过该选项，PG连接到一个工业以太网，通过S7-400站路由到一个连接的PROFIBUS子网上。

下图显示了该项设置的前提条件。

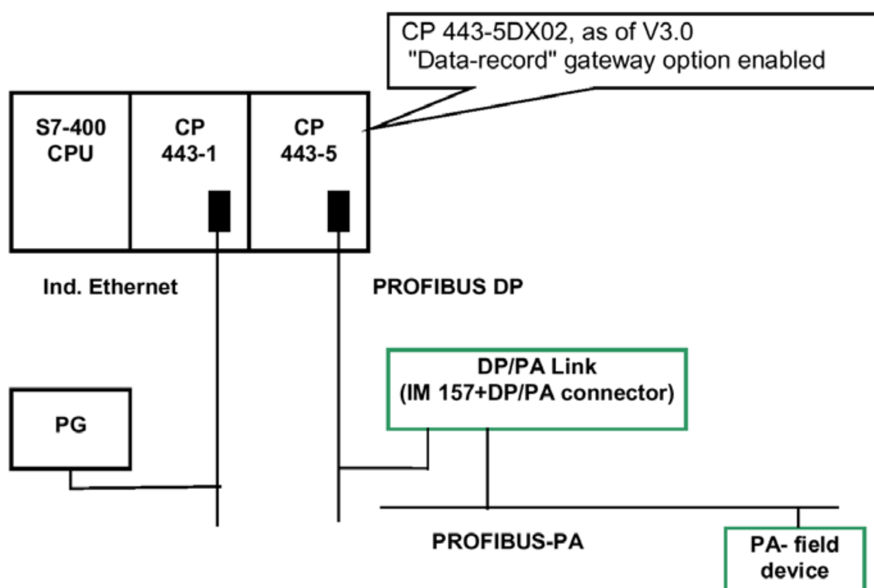
带DP/PA连接器的IM 157，用于连接PROFIBUS-PA



作为Y-link的IM 157



在工业以太网中的PG



23.6 在停机模式下进行诊断

23.6.1 判定停机原因的基本步骤

要判断CPU为什么进入“停机”模式，可按如下步骤进行：

1. 选择已进入停机的CPU。
2. 选择菜单命令**PLC > Diagnostics/Settings > Module Information**。
3. 选择“Diagnostic Buffer（诊断缓冲区）”标签。
4. 可以从诊断缓冲区中最后一项判定停机的原因。

如果出现编程错误：

1. 进入条目“STOP because programming error OB not loaded”意味着，CPU已查到一个编程错误，试图启动这个（不存在的）OB块去处理编程错误。前一条指出了实际的编程错误。
2. 选择与编程错误相关的信息。
3. 点击“Open Block”按钮。
4. 选择“Stacks（堆栈）”标签。

23.6.2 停机模式下堆栈的内容

通过评估诊断缓冲区和堆栈中的内容，你可以判定用户程序执行过程中引起故障的原因。

例如，如果由于编程错误或停机指令使CPU进入停机状态。则可用“I Stack（中断堆栈）”“L Stacks（局域堆栈）”和“Nesting Stack（嵌套堆栈）”按钮显示这些堆栈中的内容。堆栈内容给出了哪个块中的哪条指令引起CPU进入停机的信息。

B堆栈内容

B堆栈或称作块堆栈，列出了所有停机前已经被调用但未完全处理的块。

I堆栈内容

当点击“I stack（中断堆栈）”的按钮时，将显示中断点的数据。I堆栈，或称中断堆栈包含着中断时有效的数据或状态，例如：

- 累加器内容和寄存器内容
- 打开的数据块及其大小
- 状态字的内容
- 优先级（嵌套层次）

- 中断的块
- 中断后程序将继续处理的块

L堆栈内容

对于B堆栈中所列出的每个块，都可以通过选择该块并点击“L Stack（局域堆栈）”按钮显示相应的局域数据。

L堆栈，或称作局域数据堆栈，包含中断时用户程序正在处理的块的局域数据。

解释和评估所显示的局域数据需要更深入的系统知识。显示的第一部分的数据相应于块中的临时变量。

嵌套堆栈内容

当点击“Nesting Stack（嵌套堆栈）”按钮时，显示嵌套堆栈在断点处的内容。

嵌套堆栈是逻辑操作A(、AN(、O(、ON(、X(和XN(使用的存储区。

只有当中断时有括号操作仍在打开，该按钮才激活。

23.7 检查扫描循环时间以避免时间错误

在模板信息的“Scan Cycle Time（循环扫描时间）”标签中可以给出有关用户程序扫描循环时间的信息。

如果最长的循环时间接近所组态的最大扫描循环监控时间，就会存在由于循环时间的波动引起时间错误的危险。可通过延长用户程序的最大循环时间（监控时间）来避免这种危险。

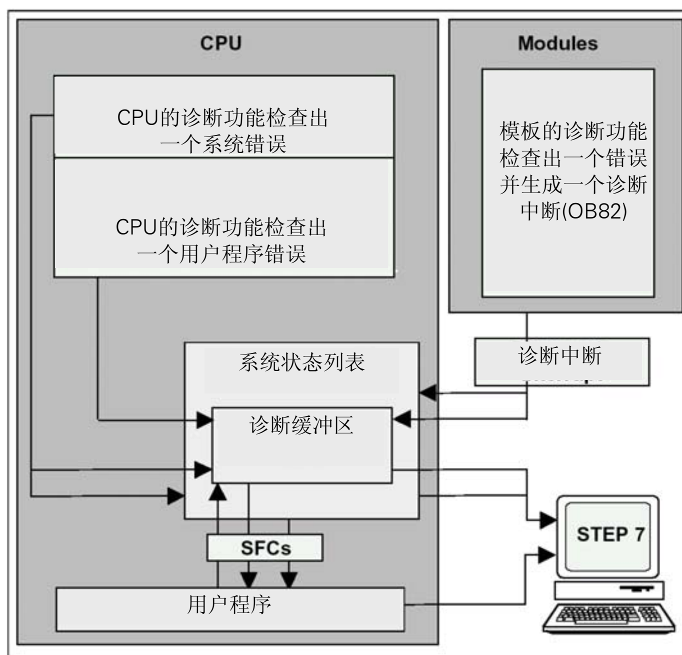
如果循环长度短于所组态的最小循环时间，则CPU/FM会自动将循环周期延长至所组态的最小循环时间。对于CPU，在这个延长时间内将处理背景OB（OB90）（如果它已下装）。

设置扫描循环时间

当组态硬件时，可以设置最大和最小循环时间。为此，双击组态列表离线视窗中的CPU/FM定义它的特性。可以在“Cycle/Clock Memory（循环周期/时钟存储器）”标签中输入适当的值。

23.8 诊断信息流向

下图所示为在SIMATIC S7中诊断信息的流向



显示诊断信息

可以在用户程序中使用SFC 51 RDSYSST读出诊断内容，或者用STEP 7以无格式文本显示诊断信息。它们可以提供以下信息：

- 错误出现的位置和时间。
- 该输入项所属的诊断事件的类型（用户定义的诊断事件、同步/异步错误、操作模式改变）。

生成过程控制组消息

CPU向诊断缓存区中写入标准诊断事件和扩展诊断事件。如果满足下列条件，它还为标准诊断事件生成一个过程控制组消息：

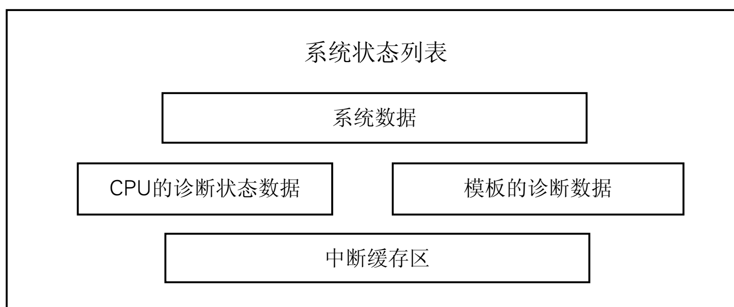
- 已指定将在STEP 7中生成过程控制消息。
- 至少有一个过程控制消息的显示单元已在CPU上登录。
- 一个过程控制组消息生成的条件是当前没有相应级别的过程控制组消息（有七个级别）。
- 每个级别可生成一个过程控制组消息。

23.8.1 系统状态列表SSL

系统状态列表（SSL）描述可编程控制器的当前状态。它提供了组态、当前参数赋值、当前CPU的状态和次序以及所属模板的信息。

系统状态列表中的数据只能读取不能修改。它是一个实际的列表，只当有请求时才会生成。

使用系统状态列表能够显示的信息可分为四个区域。



读取系统状态列表

有两种读取系统状态列表的方法：

- 间接通过编程器中STEP 7的菜单命令（例如，存储器组态、静态CPU数据、诊断缓冲、状态显示）。
- 直接在用户程序中通过系统功能SFC51 RDSYSST，输入所需部分系统状态列表的编号（参见块的帮助）。

系统状态列表的系统数据

系统数据是CPU内部数据或已分配的性能数据。下表展示可以显示的信息主题（部分系统状态列表）：

标　题	信　息
模板标识	模板的订货号、类型标识及版本
CPU特性	CPU的系统时间、系统特性（如：多CPU）以及语言描述
存储区	模板的存储器组态（工作存储器的大小）
系统存储区	模板的系统存储器（例如：存储位、定时器、计数器的数量、存储器的类型）
块类型	模板中存在哪些块（OB、DB、SDB、FC、FB），某一类型块的最大数量，以及某一类块的大小
中断及错误的赋值	中断/错误与OB之间的赋值关系
中断状态	中断处理/生成中断的当前状态
优先级的状态	由参数设定哪个OB块被执行、哪个优先级被禁止
运行模式和模式转换	可选择哪种运行模式，最后一次运行模式的变化，当前的运行模式

CPU中的诊断状态数据

诊断状态数据描述了由系统诊断所监视的组件当前状态。下表展示可以显示的信息主题(部分系统状态列表）：

标　题	信　息
通讯状态数据	当前在系统中设置的所有通讯功能
诊断模板	在CPU中登录了的有诊断能力的模板
OB的启动信息列表	有关CPU中OB的启动信息
启动事件列表	OB的启动事件及优先级
模板状态信息	所有已插入的、有故障的或生成硬件中断的模块的状态信息

模板的诊断数据

除CPU之外，其它模板也具有诊断能力（SM、CP、FM），它们的数据被存入系统状态列表中。下表所示为能够显示的信息主题（部分系统状态列表）：

标　题	信　息
模板诊断信息	模板起始地址、内部/外部故障、通道故障、参数错误（4字节）
模板诊断数据	某个特定模板的所有诊断数据

23.8.2　传送自己生成的诊断报文

通过系统功能SFC 52 WRUSMSG可以扩展SIMATIC S7的标准系统诊断：

- 在诊断缓存区中输入自己生成的诊断信息（例如，有关用户程序的执行信息）。
- 传送用户定义的诊断消息到已登录的站点（如PG、OP或TD监视设备）。

用户定义的诊断事件

诊断事件可以被分为事件级别1至F。用户定义的诊断事件属于事件级别8至B。可按如下所示分为两组：

- 事件级别8和9包括那些有固定编号和预定文本的消息，可根据编号调用。
- 事件级别A和B包括那些可由用户分配一个号码（A000至A0FF，B000至B0FF）和文本的消息。

向站点传送诊断消息

除了可以在诊断缓存区存入用户定义的条目，还可以用SFC 52 WRUSMSG将用户定义的诊断消息传送到已登录的显示设备上。当调用SFC 52，SEND=1时，诊断消息被写入发送缓冲区并自动发送到在CPU上登录的站点。

如果无法传送消息（比如，因为没有登录的显示设备或由于传送缓冲区已满），则用户定义的诊断事件仍然输入到诊断缓存区中。

生成一个带确认的消息

如果确认了一个用户定义的诊断事件并且想要记录这个确认，可按如下进行：

- 当事件到来时，事件状态将1写入一个BOOL类型的变量，当事件离开时事件状态将0写到这个变量。
- 然后使用SFB33 ALARM监视这个变量。

23.8.3 诊断功能

系统诊断将对PLC中发生的错误进行检测、评估和报告。为此，每个CPU和每个有系统诊断能力的模板（如FM354）都有一个诊断缓冲区，在这个缓冲区内诊断事件的详细信息按它们出现的顺序存入。

诊断事件

下列事项将作为诊断事件显示，例如：

- 模板上的内部和外部错误
- CPU中的系统错误
- 操作模式改变（如，从RUN到STOP）
- 用户程序中的错误
- 插/拔模板
- 用系统功能SFC52输入的用户报文

在存储器全清后诊断缓存区中的内容仍然保留。使用诊断缓冲区还可在后期对系统错误进行分析，查找停机原因并对出现的每个诊断事件分类。

获取诊断数据

不必为获取系统诊断的诊断数据而编写程序。这是一个可自动运行的标准特性。SIMATIC S7提供各种诊断功能。一些诊断功能是集成在CPU上的，另一些由模板提供（SM、CP和FM）。

显示故障

内部和外部模板故障会显示在模板的前面板上。在S7硬件手册中描述了有关LED的显示内容以及对如何对其评估的描述。在S7-300中，内部和外部故障作为一个组错误显示。

CPU能够识别系统错误和用户程序错误并将诊断信息存入到系统状态列表和诊断缓存区中。这些诊断信息可以由编程器读出。

具有诊断能力的信号模板和功能模板可检测内部和外部模板错误并产生一个诊断中断，可通过一个中断OB对其响应。

23.9 处理错误的程序

当CPU检测到程序处理过程中的错误（同步错误）和可编程控制器中的错误（异步错误）时，CPU会调用适当的组织块（OB）处理错误：

错　误	错误OB
I/O冗余错误	OB 70
CPU冗余错误	OB 72
时间错误	OB 80
电源错误	OB 81
诊断中断	OB 82
插/拔模板中断	OB 83
CPU硬件故障	OB 84
优先级错误	OB 85
机架故障或分布式I/O的站故障	OB 86
通讯错误	OB 87
编程错误	OB 121
I/O访问错误	OB 122

如果相应的OB不存在，CPU进入STOP模式(OB70、OB72、OB81、OB87除外)。此外，可在OB中调用指令以便对这种错误作出响应。这意味着可以减小或根除错误影响。

基本步骤

生成并打开OB

1. 显示CPU的模板信息。
2. 选择“Performance Data（性能数据）”标签。
3. 根据所显示的列表，确定需要编程的OB是否能在该CPU上执行。
4. 在程序的“Block（软件块）”文件夹中插入并打开该OB块。
5. 输入故障处理程序。
6. 将OB下载到可编程控制器。

处理故障的编程措施

1. 评估OB块的局部数据以判断故障的确切原因。局部数据中的变量OB8xFLTID和OB12xSWFLT中包含错误代码。其含义描述在《系统和标准功能参考手册》里。
2. 转向响应故障的程序段。

在标头为“用SFC51 （RDSYSST）作模板诊断的示例”的关于系统和标准功能的在线帮助中，能够找到处理诊断中断的示例。

在相应的块帮助中可以找有关OB、SFB、SFC的更详细的信息。

23.9.1 评估输出参数RET_VAL

系统功能用输出参数RET_VAL（返回值）指示CPU是否正确地执行了SFC功能。

返回值中的错误信息

返回值是整数类型（INT）。整数的符号位指示该整数是正还是负。返回值与数值“0”之间的关系用以指示在功能执行过程中是否有错误出现（见下表）：

- 如果功能执行时出现错误，返回值小于“0”。整数的符号位是“1”。
- 如果功能执行时没有错误，返回值大于等于“0”。整数的符号位是“0”。

由CPU处理SFC	返回值	整数的符号
错误出现	小于“0”	负（符号位是“1”）
没有错误	大于等于“0”	正（符号位是“0”）

对故障信息的反应

如果在SFC执行时出现错误，SFC在返回值（RET_VAL）中提供一个错误代码。

作以下区分：

- 所有SFC都可以输出的通用错误代码
- 依据SFC的特定功能而输出的特定错误代码

传送功能值

有些SFC还用输出参数RET_VAL来传送功能值，例如SFC64 TIMETCK用RET_VAL传送它已读出的系统时间。

在SFB/SFC帮助中可以找到一些有关输出程序SFB/SFC更详细的信息。

23.9.2 当检测到错误时故障OB的响应

可检测的错误

系统程序可以检测以下错误：

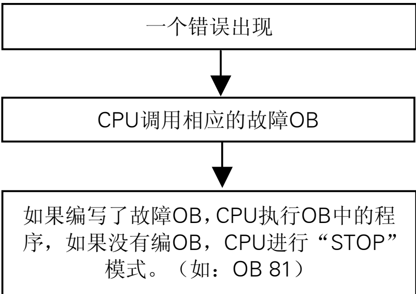
- 不正确的CPU功能
- 系统程序执行中的错误
- 用户程序中的错误
- I/O中的错误

根据错误类型的不同，CPU被设定为STOP模式或调用一个故障OB。

编程响应

可以编写程序响应不同类型的错误，并决定CPU的反应方式。为某个特定的错误而编制的

程序可以保存在一个故障OB中。如果故障OB块被调用，程序就会被执行。



故障OB

按如下所示区别同步错误和异步错误：

- 同步错误可以分配给一个MC7指令（例如，对一个已移走的信号模板使用装载指令）。
- 异步错误可分配给一个优先级或给整个可编程控制器（例如循环超时）。

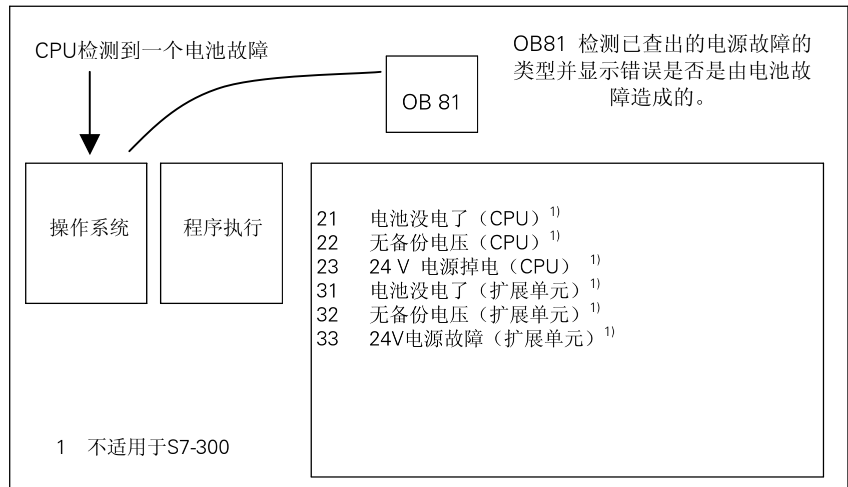
下表所示为可能出现的错误类型。参考《S7-300可编程控制器，硬件及安装手册》或《S7-400，M7-400可编程控制器，硬件及安装手册》，可找到有关CPU是否支持这些特定的OB的信息。

错误级别	错误类型	OB	优先级
冗余	I/O冗余错误(只在H CPU中)	OB 70	25
	CPU冗余错误(只在H CPU中)	OB 72	28
异步的故障	时间错误	OB 80	26
	电源错误	OB 81	（或者28如果故障OB在启动程序中调用了）
	诊断中断	OB 82	启动程序
	插/拔模板中断	OB 83	
	CPU硬件故障	OB 84	
	程序顺序错误	OB 85	
	机架故障	OB 86	
	通讯错误	OB 87	
同步	编程错误	OB 121	引起错误的OB的优先级
	I/O访问错误	OB 122	

故障OB81使用举例

使用错误OB的局域数据（启动信息），可以判断已出现的错误的类型。

例如，如果CPU检测到一个电池故障，操作系统则调用OB81（见图）。



可以编写一个程序来判定由OB81的调用而触发的事件代码。也可以编写程序作出响应，如，激活与操作站上的指示灯相连的输出。

故障OB81的局域数据

下表所示为在OB81的临时变量声明表中进行声明的临时变量（启动信息）。

符号“Battery error（BOOL）”必须被标识作为输出（例如，Q4.0），这样程序的其它部分就能够访问这些数据了。

声明	名　称	类型	描　述
TEMP	OB81EVCLASS	BYTE	错误级别/错误标识39xx
TEMP	OB81FLTID	BYTE	错误代码： b#16#21 = CPU中至少有一个备份电池没电了[1)] b#16#22 = 在CPU中无备份电压 b#16#23 = CPU中24-V电源掉电[1)] b#16#31 = 至少有一个扩展机架上的一个备份电池没电了[1)] b#16#32 = 在一个扩展机架上没有备份电压[1)] b#16#33 = 一个扩展机架24V电源掉电[1)]
TEMP	OB81PRIORITY	BYTE	优先级=26/28
TEMP	OB81OBNUMBR	BYTE	81 = OB81
TEMP	OB81RESERVED1	BYTE	保留
TEMP	OB81RESERVED2	BYTE	保留
TEMP	OB81MDLADDR	INT	保留

声明	名　称	类型	描　述
TEMP	OB81RESERVED3	BYTE	只与错误代码B#16#31，B#16#32，B#16#33相关
TEMP	OB81RESERVED4	BYTE	
TEMP	OB81RESERVED5	BYTE	
TEMP	OB81RESERVED6	BYTE	
TEMP	OB81DATETIME	DATEAND TIME	启动OB的日期和时间
1）=不适用于S7-300			

故障OB81程序举例

在STL程序举例中说明了如何在OB81中读错误代码。

程序结构如下：

- OB81中的错误代码（OB81 FLTID）被读出并与事件“电池没电”（B#16#3921）的数值作比较。
- 如果错误代码符合“（battery exhausted）电池没电”的代码，程序则跳到标号为Berr的指令并激活输出batteryerror。
- 如果错误代码与“（battery exhausted）电池没电”的代码不符，程序则将错误代码与“电池故障”的代码作比较。
- 如果错误代码符合“电池故障”代码，程序跳转到标号“Berr”，并激活输出“batteryerror”。否则该块结束。

```
    AWL                          说明
L       B#16#21                  // 比较事件代码“电池没电”
                                 // （B#16#21）
L       #OB81_FLT_ID             // 与OB81的错误代码。
==I                              // 如果相同（电池没电），
                                 // 跳转到Berr。
JC Berr
L       B#16#22                  // 比较事件代码“电池故障”
                                 //（b#16#22）
==I                              // 与OB81的错误代码。
JC BF                            // 如果不相同，跳转到 Berr。
BEU                              // 无电池故障信息
Berr：L      B#16#39             // 与下一事件ID进行比较
L       #OB81_EV_CLASS           // OB81的错误代码。
==I                              // 如果电池故障或电池没电找到，
S       batteryerror             // 设置输出“battery error”。
                                 //（符号表变量）
L       B#16#38                  // 比较ID，以确定OB81
==I                              // 错误代码事件。
```

R　　batteryerror　　　　　// 复位输出“battery error”，当错误固定时。

在相应的块帮助中可以找有关OB、SFB、SFC的更详细的信息以及事件ID解释。

23.9.3　为故障诊断插入替代值

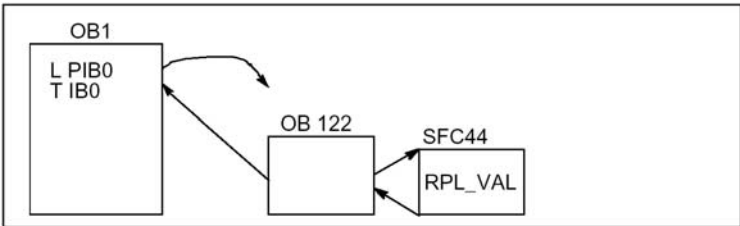
对于某种类型的故障（如，受断线影响的输入信号），可以为由于故障而无法使用的数值提供一个替代位。可用以下两种方法来提供替代值：

- 用STEP 7为可组态的输出模板分配替代值。无法得到赋值参数的输出模板用缺省替代值0。
- 用SFC44 RPLVAL，可以在故障OB中编写替代值（只适用于输入模板）。

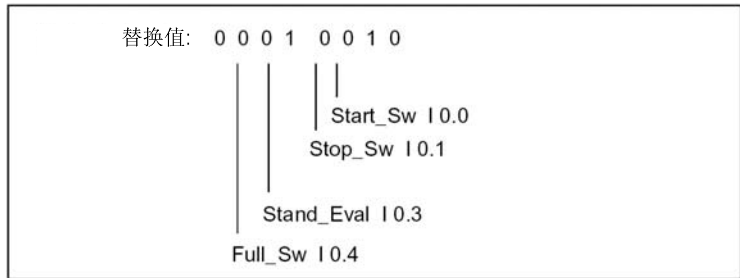
  对于所有引起同步错误的装载指令，可以在故障OB中为累加器内容指定一个替代值。

替代数值程序举例

在以下示例程序中，在SFC44 RPLVAL中有一个可用的替代值。下图说明了CPU是如何在检测到一个输入模板没有反应时调用OB122的。



在这个示例中，下图所示的替代值在程序中被输入，这样程序就可以用可行的数值继续操作。



如果一个输入模板有故障，执行指令L PIB0就会产生一个同步错误并启动OB122。作为标准，这个装载指令读得数值0。然而，用SFC44，可以为过程定义任何合适的值。SFC用指定的替代值替换累加器中的内容。

以下示例程序可写在OB122中。下表所示为在OB122的变量声明表中声明的临时变量（启动信息）。

声明	名称	类型	描述
TEMP	OB122EVCLASS	BYTE	错误级别/错误ID29xx
TEMP	OB122SWFLT	BYTE	错误代码： 16#42， 16#43， 16#44 [1]， 16#45 [1]
TEMP	OB122PRIORITY	BYTE	优先级=错误出现的OB的优先级
TEMP	OB122OBNUMBR	BYTE	122 = OB122
TEMP	OB122BLKTYPE	BYTE	错误出现的块的类型
TEMP	OB122MEMAREA	BYTE	存储区域和访问类型
TEMP	OB122MEMADDR	WORD	出错的存储器地址
TEMP	OB122BLKNUM	WORD	出错的块的号码
TEMP	OB122PRGADDR	WORD	出错指令的相对地址
TEMP	OB122DATETIME	DATEANDTIME	启动OB的日期和时间
TEMP	错误	INT	存储SFC44的错误代码
1）不适用于S7-300			

STL		说明
	L B#16#2942	为应答读I/O时出现的时间错误，将OB122的事件码与事件码
	L #OB122SWFLT	（B#16#2942）比较， 如果一样跳到“Aerr”。
	==I	为寻址错误（向一个不存在的模板作写操作）将OB122的事件代
	JC Aerr	码与事件代码（B#16#2943）作比较。如果一样跳到“STOP”。
	L B#16#2943	标号“Aerr”：将DW#16#2912（二进制10010）传送到SFC44
	<> I	（REPL_VAL）。SFC 44将该值装载到累加器1（替代由OB122
	JC Stop	调用而触发的数值）。SFC的错误代码存储在#Error。
Aerr：	CALL ”REPL_VAL"	
	VAL ：= DW#16#2912	将#Error与0比较（如果相同则OB122执行时没出错）。
	RETVAL ：= #Error	如果没有错结束块。
	L #Erro	
	L 0	“Stop”标号：调用SFC46“STP”，
	==I	将CPU转为STOP模式。
	BEC	
Stop：	CALL “STP"	“Stop”标号：调用SFC46“STP”， 将CPU转为STOP模式。

23.9.4 I/O冗余错误（OB70）

说明

对于H CPU的操作系统，如果PROFIBUS DP上出现冗余丢失（如，DP主站总线故障或DP从站接口模板故障），或者，如果带有转换I/O的DP从站变为激活的DP主站时，系统调用OB70。

在OB70中编程

必须使用STEP 7在用户程序中生成一个OB70。在生成的块中编写要在OB70中执行的程序，并作为用户程序的一部分下载到CPU。

例如，可以为如下目的使用OB70：

- 要评估OB70中的开始信息并判定哪个条件触发了I/O冗余丢失。
- 用SFC51RDSYSST判定系统的状态。（SZLID=B#16#71）。

如果出现I/O冗余错误且OB70没有编程，CPU不停机。

如果下载了OB70并且H系统不在冗余模式，则OB70在两个CPU中都被处理。H系统保持在冗余模式。

在相应的块帮助中可以找有关OB、SFB、SFC的更详细的信息。

23.9.5 CPU冗余错误（OB72）

说明

如果出现以下事件之一，H CPU的操作系统调用OB72：

- CPU冗余丢失
- 比较错误（如RAM、PIQ）
- 主从切断
- 同步错误
- SYNC子模块出错
- 刷新过程失败
- OB72被所有的处于RUN模式或起动事件后的STARTUP模式的CPU执行

在OB72中编程

必须使用STEP 7在用户程序中生成一个OB72。在生成的块中编写将由OB72执行的程序并将它作为用户程序的一部分下载到CPU。

例如，可以为以下目的使用OB72：

- 要评估OB72中的起动信息和判定哪个事件触发了CPU冗余的丢失。
- 用SFC51RDSYSST判定系统的状态。（SZLID=B#16#71）。

- 要为系统对CPU的冗余丢失作出特定的反应。

如果出现CPU冗余错误，并且OB72没有编程，CPU不会转为STOP模式。

在相应的块帮助中，可以找有关OB、SFB、SFC的更详细的信息。

23.9.6 时间错误（OB80）

说明

当有时间错误出现时CPU的操作系统调用OB80。时间错误包括以下，如：

- 超过最大循环时间
- 通过向前修改时间而跳过日时钟中断
- 处理优先级时延迟太多

在OB80中编程

必须使用STEP 7在用户程序中生成一个OB80。在生成的块中编写要在OB80中执行的程序并将它作为用户程序的一部分下载到CPU。

例如，可为以下目的使用OB80：

- 要评估OB80中的起动信息和判定哪个日时钟中断被跳过。
- 通过使用SFC29 CANTINT，可以取消已被跳过的日时钟中断而不再执行它，只有与新的时间相关的日时钟中断全被执行。

如果没有在OB80中取消跳过的日时钟中断，则第一个跳过的日时钟中断被执行，所有其它的被忽略。

如果没有编程OB80，当CPU检测到一个时间错误时转为STOP模式。

在相应的块帮助中可以找有关OB、SFB、SFC的更详细的信息。

23.9.7 电源故障（OB81）

说明

如果在CPU中或在扩展单元中有以下故障出现，CPU的操作系统调用OB81

- 24V电压
- 电池
- 完全备份

当问题消除时OB也会被调用（OB在事件到来和离开时被调用）。

在OB81中编程

必须使用STEP 7在用户程序中生成一个OB81。在生成的块中编写要在OB81中

执行的程序并将它作为用户程序的一部分下载到CPU。

例如，可以为如下目的编写OB81：

- 要评估OB81的起动信息以及判定出现哪个电源故障。
- 要查找有电源故障的机架的号码。
- 要激活操作站上的灯指示维护人员应该换电池了。

与其它的异步故障OB相反，如果没有编写OB81，CPU检测到电源故障时不会转为STOP模式。但是这个错误会存入诊断缓冲区并且前面板上相应的LED灯亮，指示错误。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

23.9.8 诊断中断（OB82）

说明

对于一个有诊断能力的模板，如果使能了它的诊断中断，当它检测到错误，以及错误消除时CPU的操作系统会调用OB82。（该OB在事件到来和离去时都会被调用）。

在OB82中编程

必须使用STEP 7在用户程序中生成一个OB82。在生成的块中编写要在OB82中执行的程序，并将它作为用户程序的一部分下载到CPU中。

例如，可为以下目的使用OB82：

- 要评估OB82的启动信息。
- 要获得与已出现的错误有关的更确切的诊断信息。

当一个诊断中断被触发时,有问题的模板自动地在诊断中断OB的启动信息和诊断缓冲区中存入4个字节的诊断数据及其起始地址，提供了错误何时出现以及出现在哪个模板上的信息。

在OB82中编写合适的程序，可以进一步地评估模板的诊断数据（哪个通道出错，出现的是哪种错误）。使用SFC51 RDSYSST可以读出模板的诊断数据，用SFC 52WRUSRMSG可以将这些信息存入诊断缓冲区。也可以发送一个用户定义的诊断消息到监控设备。

如果没有编写OB82，当诊断中断被触发时CPU转为STOP模式。

在相应的块帮助中可以找有关OB、SFB、SFC的更详细的信息。

23.9.9 插/拔模块中断（OB83）

说明

S7-400 CPU以大约1秒的间隔监视着中央机架和扩展机架上现有的模板。

在上电后，CPU检测由STEP 7生成的组态列表中所列的所有模板是否都实际地插入了。如

果所有模板都有，这个实际组态被存储并用作对模板进行循环监控的参考值。在每一扫描循环周期内，新检测到的实际组态与原来的实际组态作比较。如果发现两个组态有差异，则发出插/拔模板中断信号，并且将信息存入诊断缓冲区和系统状态列表。在RUN模式下，启动插/拔模板中断OB。

注意

电源模板、CPU和IM不能在RUN模式下移开。

在移走和插入模板两个操作之间应至少相隔两秒，以便让CPU检测到移走的或插入的模板。

对新插入的模板进行参数赋值

如果一个模板在RUN模式下插入，CPU会检测新模板的类型与原来模板是否匹配。如果匹配，对该模板赋予参数。将缺省参数或用STEP 7分配的参数传送到模板中。

在OB83中编程

必须使用STEP 7在用户程序中生成一个OB83。在生成的块中编写要在OB83中执行的程序，并将它作为用户程序的一部分下载到CPU中。

例如，可以为以下目的使用OB83：

- 要评估OB83的启动信息。
- 用系统功能SFC55至59对新插入的模板赋值参数。

如果没有编写OB83，当一个插/拔模板中断出现时，CPU从RUN转为STOP。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

23.9.10 CPU硬件故障（OB84）

说明

当CPU检测到连至MPI网络的接口故障、连至通讯总线的接口故障或连至分布式I/O网卡的接口故障时操作系统调用OB84；例如，在总线上检测到一个不正确的信号电平。故障消除时也会调用该OB块（事件到来和离开时都调用该OB）。

在OB84中编程

必须使用STEP 7在用户程序中生成一个OB84。在生成的块中编写要在OB84中执行的程序并将它作为用户程序的一部分下载到CPU。

可以为以下目的使用OB84：

- 要评估OB84的起动信息。
- 用系统功能SFC52 WRUSMSG发送消息至诊断缓存区。

如果OB84没有编程，当检测到CPU硬件错误时CPU转为STOP模式。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

23.9.11 编程顺序错误（OB85）

说明

CPU的操作系统调用OB85：

- 当一个中断OB的启动事件存在，但该OB块由于没有下载到CPU而不能被执行时。
- 当访问一个系统功能块的背景数据块时出错。
- 当刷新过程映像区时出错（模板不存在或出故障）。

在OB85中编程

必须使用STEP 7在用户程序中生成一个OB85。在生成的块中编写要在OB85中执行的程序并将它作为用户程序的一部分下载到CPU。

例如，可以为以下目的使用OB85：

- 要评估OB85的启动信息和判定哪个模板损坏或没插入（指定模板的起始地址）。
- 用SFC49 LGCGADR查找相关模板所在的槽。

如果没有编写OB85，当优先级错误被检测到时CPU转为STOP模式。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

23.9.12 机架故障（OB86）

说明

当检测到下列故障时，CPU的操作系统调用OB86：

- 中央扩展机架故障（不适用S7-300找不到IM或IM损坏，或者），例如连接电缆断线、机架上的分布电源故障等。
- PROFIBUS-DP 主站、从站故障或PROFINET 网络中IO元件或IO系统故障。

故障消除时也会调用该OB块（事件到来和离开时都调用该OB）。

在OB86中编程

必须使用STEP 7在用户程序中生成一个OB86。在生成的块中编写要在OB86中执行的程序，并将它作为用户程序的一部分下载到CPU中。

例如，可以为以下目的使用OB86：

- 要评估OB86的启动信息和判定哪个机架损坏或找不到。
- 用系统功能SFC 52 WRUSNSG将消息存入诊断缓冲区，并发送消息到监视设备上。

如果没有编写OB86，当检测到机架故障时CPU转为8TOP模式。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

23.9.13 通讯错误（OB87）

说明

当使用通讯功能块或全局数据通讯进行数据交换时出现通讯错误，CPU的操作系统调用OB87，例如：

- 当接收全局数据时，检测到不正确的帧标识。
- 用于全局数据状态信息的数据块不存在或太短出。

在OB87中编程

必须使用STEP 7在用户程序中生成一个OB87。在生成块中编写要在OB87中执行的程序，并将它作为用户程序的一部分下载到CPU。

例如，可以为以下目的使用OB87：

- 要评估OB87的启动信息。
- 如果找不到用于全局数据通讯状态信息的数据块，要生成该数据块。

如果OB87没有编程，当检测到通讯错误时CPU转为STOP模式。

在相应的块帮助中可以找有关OB、SFB、SFC的更详细的信息。

23.9.14 编程错误（OB121）

说明

当出现编程错误时，CPU的操作系统调用OB121，例如：

- 寻址的定时器不存在。
- 调用的块未下载。

在OB121中编程

必须使用STEP 7在用户程序中生成一个OB121。在生成的块中编写要在OB121中执行的程序，并将它作为用户程序的一部分下载到CPU中。

例如，可以为以下目的使用OB121：

- 要评估OB121的启动信息。
- 要在消息数据块中存入故障原因。

如果不编写OB121，当检测到编程错误时CPU转为STOP模式。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

23.9.15 I/O访问错误（OB122）

说明

当STEP 7指令访问一个信号模板的输入或输出时，而在最近的一次暖起动中没有分配这样的模板，CPU的操作系统会调用OB122，例如：

- 直接访问I/O出错（模板损坏或找不到）
- 访问一个CPU不能识别的I/O地址

在OB8122中编程

必须使用STEP 7在用户程序中生成一个OB8122。在生成的块中编写要在OB122中执行的程序，并将它作为用户程序的一部分下载到CPU中。

可以为以下目的使用OB122：

- 要评估OB122的启动信息。
- 要调用系统功能SFC44为输入模板提供一个替代值以便使程序可以用这个有意义的、随过程变化的数值继续执行。

如果不编写OB122，当检测到I/O访问错误时CPU转为STOP模式。

在相应的块帮助中可以找到有关OB、SFB、SFC的更详细的信息。

24 打印与归档

24.1 打印项目文档

自动化任务的控制程序编制完成后，便可以利用集成于STEP 7中的打印功能将所有重要数据打印出来，作为项目的文献资料。

项目中可打印的部分

可以从SIMATIC Manager直接打印选定对象的内容，也可以打开相关的对象再起动打印过程。

一个项目中的下列部分可通过SIMATIC Manager直接打印出来：

- 对象的树形图（项目或库的结构）
- 对象清单（一个对象文件夹的目录）
- 对象内容
- 信息

通过打开相关的对象，一个项目中的下列部分可以打印出来：

- 以梯形图、语句表或功能图表述的程序模块，也能以其它语言表述（选件）
- 带有符号名称和绝对地址的符号表
- 组态列表，包括可编程控制器中模块的排列及模块参数
- 诊断缓冲器区的内容
- 变量表，包括监控格式及监控值和更改值
- 参考数据，例如：交叉表、赋值表、程序结构、未使用地址表、无符号地址表
- 全局数据表
- 带有模块状态的模块信息
- 操作员相关文本（用户文本和文本库）
- 可选软件包的文档，例如其它编程语言

DOCPRO选件软件包

可以使用选件软件包DOCPRO，生成、编辑和打印标准化的接线手册。该软件包可生成符合DIN和ANSI标准的设备文档。

24.1.1 打印的基本步骤

打印的步骤如下：

1. 打开相关的对象，在屏幕上显示要打印的信息。
2. 在应用窗口上使用菜单命令**File > Print** 打开“打印”对话框。根据使用的应用程序，菜单条上的第一项可能不是“File”，而是应用程序要处理的对象，如“Symbol Table（符号表）”。
3. 在打印对话框中，如有必要可以修改打印选项（打印机，打印范围，份数等），然后关闭对话框。

有些对话框有“打印”按钮，例如“模块信息”对话框。点击该按钮即可打印对话框的内容。

不需要打开程序模块。可在SIMATIC Manager中使用菜单命令 **File > Print** 直接打印。

24.1.2 打印功能

打印对象所具有的附加功能见下表：

打印对象	**菜单命令**	**功能**	**功能**	**功能**
		打印预览	页面设定 “纸张大小”	打印设定 “页眉和页脚”
程序模块、STL源文件	File > *	•	•	•
模板信息		—	•	•
全局数据表	GD Table> *	•	•	•
组态列表	Station > *	•	•	•
对象、对象文件夹	File > *	—	•	•
参考数据	Reference Data > *	•	•	•
符号表	Symbol Table > *	•	•	•
变量表	Table> *	—	•	•
连接表	网络> *	•	•	•
操作员相关文本（用户文本，文本库）	Texts> *	•	•	•
* ： *号是通配符，表示菜单命令中的相关功能（例如，打印预览或页面设定）				

对于每个单独的打印对象的逐步引导说明，可以在“How to Print（如何打印）”之下找到。

打印预览

利用“打印预览”功能，可以显示需要打印文档的页面布局。

注意

文档最终的打印格式在打印预览中不能显示。

设定页面格式以及页眉和页脚

使用菜单命令File > Page Setup，可以对要打印的所有文档设定纸张大小（例如A4、A5、信封）以及打印方向（横向、纵向）。可以对当前段或整个文档分别设置。

调整文档的页面布局使其符合打印纸的格式。若文档过宽，右侧的页边会打印到后续页上。

如果选择带有页边的页面格式（如A4带页边），则打印出来的文献在页的左侧留出一个页边，可以穿孔和装订。

选择“Labeling Fields(标签区)”，可以为整个打印文档或打印当前段设定页眉和页脚。

24.1.3 关于打印对象树形图的特别说明

在“打印对象清单”对话框中，除了对象清单之外，还可以通过选中“树形图”选项而打印出对象的树形图。

如果在“打印范围”中选择“全部”，则可以打印出整个树形图。若选择“选择”，则可以打印出被选中对象以下的树形图。

注意

对话框中的设定仅适用于打印清单和树形图，而不适应于打印对象的内容；打印内容时在相应的应用中有相关的设定。

24.2 项目和库的归档

可以将项目或库以压缩的形式存储在一个归档文件中。这种压缩存储过程可以在硬盘上进行也可以在一种便携的数据载体上进行（例如，用软盘）。

归档程序

在STEP 7中，使用归档功能可以对需选程序进行归档。归档应用程序ARJ和PKZIP 4.0作为一个组成部分已包含在STEP 7软件包内。在...\Step7\S7bin\路径下可以找到这些程序及其使用说明。

如果使用下列归档程序，则应选择下列的版本（或最高版本）：

- PKZip Commandline V4.0（包含在STEP 7中）
- WinZip 版本6.0以上
- JAR 版本1.02以上

- ARJ V2.4.1a（只适合于可恢复的归档，包含在STEP 7中）
- ARJ32 V3.x（只适合于可恢复的归档）
- LHArc　版本2.13以上（只适合于可恢复的归档）

归档程序

STEP 7 V5.2只支持PKZip 4.0、JAR和WinZip。但上面列出的其它程序可以支持对其恢复。

如果在更早版本的STEP 7 中，只能使用ARJ32 V3.x创建归档文件，恢复归档文件时只能使用相同的程序。

使用PKZIP V4.0创建归档程序时，网络驱动器所需的时间比本地驱动器所需的时间要长些。

24.2.1　使用保存/归档

另存为（Save As）

使用另存功能，可以创建项目副本并以其它名称保存。

可以利用此功能：

- 生成备份副本。
- 复制一个存在的项目以便修改后用于其它目的。

生成副本的最快方法是，选择“Save as”选项并且不重新安排对话框。项目目录中的整个文件结构都不作任何检查地被复制，然后以其它名称保存。

数据介质必须有足够的空间来存储备份副本。不要尝试将项目保存到磁盘，因为磁盘通常没有足够的可用空间。要往磁盘上传输项目数据，使用“归档”功能。

带有重新排列任务的保存将花费较长的时间，当对象不能被复制和保存时会显示出一条信息。引起的原因可能是一个对象所需的选项软件包或数据不完全。

归档

可以将项目或库以压缩的形式存储在一个归档文件中。这种压缩存储过程可以在硬盘上进行，也可以在一种便携的数据载体上进行（例如，用软盘）。

将项目转移到软盘只能用归档文件的形式。若一个项目过大，则在选用归档程序时注意其应具有生成多张软盘上连续归档文件的功能。

被以压缩形式存入一个归档文件的项目或库不能被编辑修改。若希望对它们进行编辑，就必须将数据解压缩，即恢复项目或库。

24.2.2　对归档的要求

为了将一个项目或一个库进行归档，必须满足以下要求：

- 必须在系统中安装归档程序，归档程序与STEP 7的连接参见在线帮助中的“归档/恢复

的步骤”。

- 所有与项目有关的数据无一例外地必须在项目目录之内或项目子目录之内。当使用C语言开发环境时，有可能将数据存储在其它位置。则归档文件中不包括这些数据。
- STEP 7 V5.2只支持归档程序Pkzip 4.0，JAR。但是在恢复时也支持ARJ和LHArc程序。

24.2.3 归档/恢复的步骤

可以对项目或库进行归档和恢复，使用的菜单命令为**File > Archive**或**File > Retrieve**。

注意

不能对压缩到归档文件中的项目或库进行编辑。如果希望再次编辑它们，必须提取数据，即恢复项目或库。

当进行恢复操作时，项目或库会自动恢复到项目/库清单之中。

设定目标目录

可在SIMATIC Manager中使用菜单命令**Options > Customize**设定目标目录。

在该对话框的“Archive”中，可以选择或取消“恢复时检查目标路径”选项。

如果绍选项无效，那么在同一个对话框的“通用”标签中为“项目存储位置”和“库存储位置”而设置的路径将用来作为恢复的目标路径。

将一个归档文件复制到软盘

可以将一个项目/库进行归档，然后将归档文件复制到软盘上。也可以在“Archive（归档）”对话框中选择软盘驱动器作为目标路径。

25 使用M7可编程控制系统

25.1 M7 系统程序

具有标准PC机结构的M7-300/M7-400自动控制微机，在SIMATIC自动化平台上，使自由编程功能得到扩展。可以使用高级语言例如C语言或图形化编程语言CFC（连续功能图）来编制SIMATIC M7的用户程序。

为了生成程序，除STEP 7之外，你还需要用于M7-300/400的系统软件M7-SYS RT和用于M7程序的开发环境（Pro C/C++或CFC）。

基本步骤

当使用SIMATIC M7创建一个自动控制方案时，需要完成一系列基本任务。下表指出了对于大多数项目都要执行的任务，并将其定义为基本步骤。下表还给出了在本手册或其它手册中可供参考的相关章节。

步　骤	说　明
设计自动化控制方案	M7专用； 参考：M7-SYS RT编程手册
启动STEP 7	同S7
创建项目结构 设置站点 建立硬件组态	同S7
建立通讯连接组态	同S7
定义符号表	同S7
生成C语言或CFC用户程序	M7专用； 参考：ProC/C++
配置操作系统 在M7-300/M7-400中安装操作系统 向M7下载硬件组态和用户程序	M7专用； 参考：M7-SYS RT用户手册
检测并调试用户程序	ProC/C++
运行监控和M7诊断	同S7，但是没有用户自定义的诊断
打印与归档	同S7

在M7中有什么不同？

对于M7-300/M7-400，STEP 7不支持下列功能：

- 多CPU运算—若干个CPU的同步操作
- 变量强置

- 全局数据通讯
- 用户自定义的诊断

M7可编程控制系统的管理

STEP 7为使用M7可编程控制系统提供如下特别的支持：

- 在M7-300/M7-400中安装操作系统
- 通过编辑系统文件对操作系统进行配置
- 向M7-300/M7-400装载用户程序
- 软件升级

为了访问M7可编程控制系统的管理器，在包含有M7 CPU或FM模块工作站的项目中，选中M7程序文件夹，选择菜单命令：

PLC > Manage M7 System

在M7-SYS RT用户手册和在线帮助中有详细的说明。

25.2 用于 M7 编程的选装软件

M7选装软件

STEP 7提供了所需的如下基本功能：

- 项目的生成与管理
- 对硬件进行配置和参数设定
- 配置网络和连接
- 符号数据管理

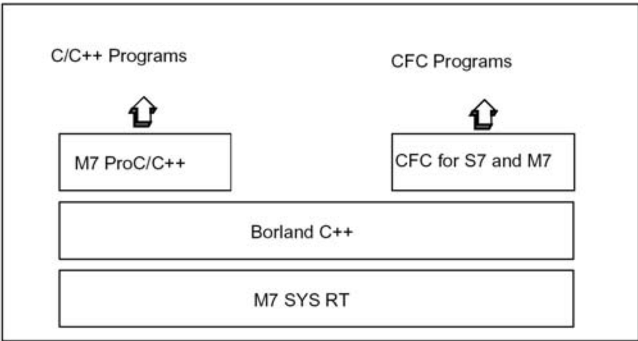
不论是使用SIMATIC S7还是SIMATIC M7，都能提供这些功能。创建M7应用程序时，除了STEP 7之外，还需要M7选装软件。

软件	内　容
M7-SYS RT	• M7 RMOS32操作系统 • M7-API系统库 • 对MPI的支持
用于S7和M7的CFC	CFC（连续功能图）编程软件
软件	**内　容**
M7-ProC/C++	• 用于在STEP 7中实现Borland开发环境的连接 • 符号的编辑与生成 • xdb386高级语言调试工具的推理规则
Borland C++	Borland C/C++开发环境

结合M7选装软件，STEP 7还可以支持下列附加功能：

- 通过多点接口（MPI）向M7可编程控制系统下装数据
- 查询有关M7可编程控制系统的信息
- 在M7可编程控制系统上进行特别设定和对M7进行复位

下图显示了M7编程对M7选装软件的依赖性。



小结

为生成...	所需的M7选装软件...
C/C++程序	• M7-SYS RT • M7-ProC/C++ • Borland C++
CFC程序	• M7-SYS RT • 用于S7和M7的CFC • Borland C++

不同软件提供的支持

为M7应用所特需的工具，一部分集成于STEP 7之中，另一部分是M7的选装软件。

下表指明了不同软件包支持的功能：

软件	提供的支持
STEP 7	• 安装M7操作系统 • 管理M7可编程控制系统 • 下载、起动和删除M7程序 • 显示状态和诊断数据 • CPU复位
M7-SYS RT	M7操作系统和M7系统软件应用帮助实现以下功能： • 程序处理的控制 • 内存和资源的管理 • 对计算机硬件和SIMATIC硬件的访问 • 处理中断 • 诊断 • 状态监控 • 通讯
M7-ProC/C++	• 集成代码的生成(将Borland的开发环境集成于STEP 7 中) • 将项目符号连接到源代码 • 集成的调试功能
Borland C++	• 生成C和C++程序
用于S7和M7的CFC	• 生成、检测和调试CFC程序 • 起动并运行CFC程序

25.3 M7-300/M7-400 操作系统

对于用高级语言C和C++生成的应用程序，操作系统的作用是非常重要的。对于这些应用，操作系统要承担下列任务：

- 访问硬件
- 管理资源
- 系统集成
- 与系统中其它部件进行通讯

为了完成自动控制任务，SIMATIC M7自动控制计算机使用M7 RMOS32（实时多任务操作系统）。M7 RMOS32经过扩展后包含了一个调用接口，M7 API（应用程序接口）将其集成到SIMATIC系统之中。

实时操作系统M7 RMOS32是用于实时和多任务控制方案的，在时间准则上使用32位处理。对于M7模块它可以有如下列配置：

- M7 RMOS32
- M7 RMOS32带MS-DOS

M7可编程控制系统的操作系统配置取决于所用的M7模块：

操作系统配置	模块/主内存	PROFIBUS-DP和TCP/IP有/无	安装在海量存储器上
M7 RMOS32	FM 356-4 / 4 MB FM 356-4 / 8 MB CPU 388-4 / 8 MB FM 456-4 / 16 MB CPU 488-3 / 16 MB CPU 486-3 / 16 MB	无 有 有 有 有 有	存储卡≥4MB或在硬盘上
M7 RMOS32带MS-DOS	FM 356-4 / 8 MB CPU 388-4 / 8 MB FM 456-4 / 16 MB CPU 488-3 / 16 MB CPU 486-3 / 16 MB	无 无 有 有 有	存储卡≥4MB或在硬盘上

26 提示与技巧

26.1 更换组态列表中的模板

如果使用HW Config修正一个站的组态，并且需更换一个新模板，应如下进行：

1. 使用拖放功能，将模板从Hardware Catalog（硬件目录）窗口中拖到旧模板上。
2. 放下新模板。应尽可能地使新模板使用已插模板的参数。

该方法比删除旧模板、插入新模板并对其赋值参数的速度快。

可以使用菜单命令**Options > Settings**（“Enable Module Swapping（使能模板交换）”），在HW Config中关闭或打开该功能。

26.2 具有大量网络站的项目

如果逐个组态所有站，并使用菜单命令**Options > Configure Network**调用NetPro，以便组态连接，站将自动放置在网络视图中，其缺点是必须根据拓扑条件安排站和子网。

如果项目中包括大量的网络站，并且需要在这些站之间组态连接，则应在网络视图中组态系统结构，并且保存概览。

1. 在SIMATIC Manager中创建新项目（菜单命令**File > New**）。
2. 启动NetPro（菜单命令**Options > Configure Network**）。
3. 在NetPro中一个站一个站地创建：
   - 使用拖放功能，从Catalog窗口中拖出站。
   - 双击站点，启动HW Config。
   - 使用拖放功能，在HW Config中放置具有通讯功能的模块（CPU、CP、FM、IF模块）。
   - 如果需要网络连接这些模板，双击组态表中的相应行，创建新的子网，并网络连接接口。
   - 保存组态，并切换到NetPro。
   - 在NetPro中，定位站和子网（使用鼠标移动对象，直至到达所需位置）。
4. 在NetPro中组态连接，根据需要校正网络连接。

26.3 重新排列

若在使用STEP 7过程中，出现一些无法解释的问题，通常可以通过对项目或库的数据库进行重新排列予以解决。

选择菜单命令**File > Rearrange**可进行重新排列。该功能可消除由于某些内容被删除后而产生的数据存储不连续，即使得项目/库所需的存储器量减少。

该功能对项目或者库的数据存储进行优化，其方法类似硬盘碎片整理程序对硬盘的文件存储进行优化。

重新排列处理的时间长短取决于要移动的数据量大小，很可能要花些时间。因此该功能不能自动执行（例如，当关闭一个项目时无法执行该功能），而应由用户在打算重新排列项目或库时进行调用。

要求

项目和库需要进行重新排列时，必须保证其中不能有正在被其它用户编辑的对象，因为这时的存取权会被封锁。

26.4 如何在多个网络中编辑符号

可以用LAD/STL/FBD程序编辑器查看和编辑多个网络中的符号。

1. 点击选择一个网络名(例如“Network 1”)
2. 按住CTRL键可同时选择其它网络
3. 右击调用文本菜单命令**编辑符号**

可以使用快捷键CTRL+A选择一个程序块中的所有网络，然后选中网络名。

26.5 用变量表进行测试

监视和修改变量表中的变量时，请注意以下编辑技巧：

- 可以在“符号”栏输入符号和地址，也可以在“地址”栏输入符号和地址。然后输入便会自动地在正确的栏中写入。
- 需要显示修改值，应将“Monitoring（监视）”触发点设在“Beginning of Scan Cycle（扫描循环开始）”，将“Modifying（修改）”触发点设在“End of Scan Cycle（扫描循环结束）”。

- 如果将光标放在红色的行上，可以显示一个简短信息来说明错误的原因。按F1键，可获得消除这些错误的建议。
- 只能输入已在符号表中定义过的符号。必须完全按照符号表中的定义来输入符号。有特殊字符的符号名必须用引号括起来（例如，“Motor.Off”，“Motor+Off”，“Motor-Off”）。
- 可以在“Online（在线）”标签（“Customize（自定义）”对话框）中关闭警告。
- 改变连接时无需事先断开连接。
- 监控变量时，可以定义监控触发器。
- 通过选择行，并执行“Force（强制）”功能，可以修改所选变量。只有高亮显示的变量才能修改。
- 无确认退出：
  在“Monitoring（监视）”、“Modifying（修改）”、“Release PQ（释放PQ）”时，如果按下ESC键，将不会询问是否需要退出就会中止“Monitoring（监视）”和“Modifying（修改）”。
- 输入一个“Contiguous Address Range（连续的地址范围）”：
  使用菜单命令**Insert > Range of Variable**。
- 显示和隐藏栏：
  使用以下菜单命令可以显示和隐藏单个栏：
  符号：**View > Symbol**
  符号注释：**View > Symbol Comment**
  状态值的格式：**View > Display Format**
  变量的状态值：**View > Status Value**
  变量的修改值：**View > Modify Value**
- 同时修改表中几行的显示格式：
  1 按住鼠标左键在所需要的表中拖动，选中那些要改变显示格式的区域。
  2 使用菜单命令**View > Select Display Format**，选择输出格式。只有那些表中允许改变格式的行才能改变格式。
- 通过F1键显示输入示例：
  - 如果将光标放在地址栏并按F1键，可以获得地址输入的示例。
  - 如果将光标放在修改值栏并按F1键，可以获得修改/强制值输入的示例。

26.6 用程序编辑器修改变量

在程序编辑器中，可以将按钮设置为二进制输入和存储器位，这样可以通过点击鼠标快速、方便地修改这些地址。

需求

- 在符号表中，已经通过菜单命令 Special Object Properties > Control at Contact对所要修改的地址分配了属性。
- 已经在LAD/STL/FBD程序编辑器的“General”表中选择了“Control at Contact”选项(菜单命令Options > Customize)。
- 已经选择了菜单命令Debug > Monitor。

触发条件是“permanent/at the cycle start（永久/周期启动）”。

只要持续按下按钮，就可以一直监视所输入的实际变量。也可以通过多重选择(CTRL键)修改多个输入。

按下按钮将对位存储器或不能修改的输入状态置1。当通过文本菜单或在变量表中输入明确的要求，或如果用STEP 7程序对地址复位，才能将其置为0。

对于非负输入（常开）或位存储器，按下按钮将使修改值为“1”；对于负的输入（常闭）或位存储器，则修改值为“0”。

使用WinCC时的注意事项

如果已经通过操作员控制和变量监视在WinCC中启动程序编辑器，只有WinCC的控制选项是允许的。否则，如果操作员已经具有WinCC的“维护权”，两个修改选项都可以使用。

编辑“Controlat Contact”属性。

26.7　虚拟工作存储器

引起STEP 7出问题的另一个原因有可能是虚拟工作存储器不够。

为了使用STEP 7，应该调整虚拟存储器的设定。调整的步骤如下：

1. 打开“控制面板”，利用命令**Start > Settings > Control Panel**并双击“System（系统）”图标。对于XP，通过**START > Desktop > Properties > Advanced > System Performance > Settings**打开。
2. 在Windows2000下，选择“Advanced”并点击“System Properties Options（系统性能选项）”按钮。在Windows XP/Server 2003下，在“System Settings（系统设定）”对话框中选择“Advanced”。
3. 点击“Change（修改）”按钮。
4. 在“最小”处输入至少40MB，在“最大”处至少150MB。

注意

对于在硬盘（默认为C：）上且为动态的虚拟存储器，必须确保为子目录TMP或TEMP提供足够的存储器量（大约20至30MB）

- 如果S7项目与虚拟存储器设定在同一分区内，则应保证有大约两倍于S7项目大小的存储器空间。
- S7项目存于其它分区，此项要求无关。

A 附录

A.1 操作模式

A.1.1 操作模式和模式转换

操作模式

操作模式描述了 CPU 在某个特定的时间点的状态。在编程启动、测试控制器和故障诊断时了解 CPU 的操作模式是有用的。

S7-300 和 S7-400 可采取以下操作模式：

- STOP (停机)
- STARTUP (启动)
- RUN (运行)
- HOLD (保持)

在 STOP 模式，CPU 检查所有组态模板或由缺省地址设置的模板是否实际存在，并且将 I/O 设置为预定义的初始状态。在 STOP 模式下用户程序不执行。

在 STARTUP 模式下，要区别启动类型“warm restart(暖启动)”“cold restart(冷启动)”和“hot restart(热启动)”：

- 在暖启动中，程序处理从头开始，使用系统数据和用户地址区的初始设置(非记忆的定时器、计数器和位存储被复位)。
- 在冷启动中，读入过程映像输入表并且STEP 7用户程序从OB1的第一条指令开始处理(也适用于暖启动)。
  - 所有由SFC在工作存储器中生成的数据块都被删除；保留下来的数据块具有来自装载存储器的预设值。
  - 过程映像区和所有定时器、计数器及位存储被复位，无论它们是否是可记忆的。
- 在热启动中，程序从中断的断点处继续运行(定时器、计数器和位存储不复位)。热启动只在S7-400 CPU上是可能的。

在 RUN 模式下，CPU 执行用户程序，更新输入和输出，处理中断和过程故障信息服务。

在 HOLD 模式，用户程序的执行被暂停，可以单步地测试用户程序。只有当使用编程器进行测试时才有可能处于 HOLD 模式。

在所有这些模式中，CPU 可以通过多点接口(MPI)进行通讯。

其它操作模式

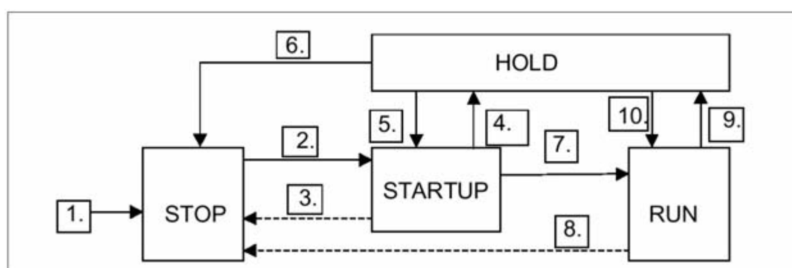
如果 CPU 尚未作好操作准备，它可以处于以下几种模式：

- off，即电源关断。
- 故障，即有故障出现。

要检测 CPU 是否有故障，将 CPU 切换到 STOP，关断电源再通电。如果 CPU 启动，打开诊断缓冲区并对问题进行分析。如果 CPU 不启动，则需要换新的了。

操作模式转换

下图所示，S7-300 和 S7-400 CPU 的操作模式及模式转换。



下表所示的操作模式转换的条件。

转换	描 述
1.	在接通电源后，CPU处于STOP模式
2.	CPU转为STARTUP模式： • 用钥匙开关或编程器将CPU转为RUN或RUN-P后 • 由通电自动触发起动之后 • 如果执行了RESUME或START通讯功能 在以上的后两种情况下，钥匙开关必须在RUN或RUN-P
3.	CPU转回STOP模式，当： • 在启动过程中检到错误 • 由钥匙开关或由编程器将CPU转为STOP • 在启动OB中执行了停机命令 • 执行了STOP通讯功能
4.	在启动程序中遇到断点，CPU转为HOLD模式
5.	在启动程序中设置了断点并且执行了"EXIT HOLD"命令（测试功能），CPU转为STARTUP模式
6.	CPU转为STOP模式，当： • 用钥匙开关或通过编程器将CPU转为STOP • 执行了STOP通讯命令
7.	如果启动成功，CPU转为RUN

转换	描 述
8.	CPU转回STOP模式当： • 在RUN模式下检测到错误且相应的OB块没有装载 • 用钥匙开关或编程器将CPU转为STOP • 在用户程序中执行了STOP命令 • 执行了STOP通讯功能
9.	当用户程序中遇到断点，CPU转为HOLD模式
10.	当设置了断点且执行了”EXIT HOLD”命令时，CPU转为RUN模式

操作模式优先级

如果同时有多个模式转换请求，则高优先级的操作模式被选中。例如，模式选择开关设为RUN，试图在编程器上将 CPU 设为 STOP，因为这一模式具有高优先级，所以 CPU 转为STOP。

优先级	模式
最高	STOP
	HOLD
	STARTUP
最低	RUN

A.1.2 STOP 模式

在 STOP 模式下用户程序不执行。所有输出都设为适当的值以保证控制过程处于安全状态。CPU 作以下检查：

- 是否有硬件问题(例如，模板故障)?
- 是否将缺省设置用于CPU或是否有参数设置?
- 编程的启动特性的条件是否满足?
- 系统软件是否有问题？

在 STOP 模式下，CPU 还可以接收全局数据并可进行被动的单向通讯，实现这些通讯需要对已组态的连接使用通讯 SFB 并且对没有组态的连接使用通讯 SFC。

存储器复位

CPU 存储器可在 STOP 模式下被复位。可以用钥匙开关(MRES)或通过编程器(例如，在下载用户程序前)手动复位存储器。

复位 CPU 的存储器使 CPU 回到它的初始状态：

- 在工作存储器和RAM装载存储器中所有的用户程序以及所有的地址区都被清除。
- 系统参数及CPU和模板的参数被复位为缺省设置。而复位前设置的MPI参数仍保留。
- 如果插入了存储卡(Flash EPROM)，CPU将存储卡中的用户程序复制到工作存储区(如

果存储卡中有适当的组态数据，则所复制的数据包括CPU和模板参数)。

诊断缓存区、MPI 参数、时间以及 CPU 运行时间计数器数器都不复位。

A.1.3 STARTUP 模式

在 CPU 启动用户程序处理之前，必须先执行起动程序。通过在起动程序中编写起动 OB，可以为循环程序指定初始设置。

有三种起动类型：暖起动、冷起动和热起动。只有 S7-400 CPU 有热起动。这必须用 STEP 7 直接在 CPU 参数中设置。

STARTUP 模式的特性如下：

- 处理起动OB中的程序(OB100暖起动，OB101热起动，OB102冷起动)。
- 不能执行时间驱动OB或过程驱动程序。
- 定时器被更新。
- 运行时间计数器数器起动运行。
- 禁止信号模板的数字输出(可通过直接访问置位)。

暖起动

除非系统要求存储器复位，否则暖起动总是允许的。在以下情形之后，暖起动是唯一可能的选项：

- 存储器复位
- 在CPU位于STOP模式时下载用户程序
- I堆栈/B截栈溢出
- 终止暖起动(由于掉电或改变模式选择开关的设置)
- 在热起动前的中断超过了选择的时间限制。

手动暖起动

以下可以触发手动暖起动：

- 模式选择开关
  (CRST/WRST 开关——如果有——必须设置为 CRST)
- 编程器上相应的命令或通讯功能。
  (如果模式选择开关设为 RUN 或 RUN-P)

自动暖起动

自动暖起动在上电后可由以下情形触发：

- 电源掉电时CPU不在STOP模式。

- 模式选择开关设为RUN或RUN-P。
- 上电后没有编程自动热起动。
- 在暖起动过程中CPU被电源掉电中断(与编程的起动类型无关)。

CRST/WRST 对自动暖起动没有影响。

没有后备电池的自动暖起动

如果 CPU 在没有后备电池的情况下运行(免维护操作是必要的)，在电源接通后或电源从关电到上电后，CPU 存储器自动复位并执行暖起动。用户程序必须装载到一个 Flash(闪存)EPROM(存储卡)。

热起动

CPU 处于 RUN 模式掉电，随着电源恢复，S7-400 CPU 运行完整的初始化例行程序，然后自动执行一个热起动。在热启动过程中，用户程序从它被中断的地方继续执行。掉电前没能执行完的用户程序部分就是剩余循环。剩余循环也可以包含时间驱动和中断驱动程序部分。

只有在 STOP 模式下没有进行用户程序的修改(如，重新装载修改过的块)以及没有其它理由作暖起动时才允许作热起动。手动和自动热起动都可以。

手动热起动

只有当 CPU 中设置了适当的参数并且由以下原因造成停机时才有可能执行手动热起动。

- 模式选择开关从RUN转化STOP。
- 用户编写的STOP程序或在调用了未装载的OB后停机。
- 通过编程器或通讯功能的命令引起的STOP模式。

手动热起动可由以下事件触发。

- 模式选择开关
  CRST/WRST 必须设为 WRST。
- 编程器或通讯功能中相应的命令(模式选择开关设为RUN或RUN-P)
- 当CPU参数中设置了手动热起动。

自动热起动

在以下情形下电源上电可触发自动热起动：

- 电源掉电时CPU不在STOP或HOLD模式。
- 模式选择开关设为RUN或RUN-P。
- 在CPU参数中设置了上电后执行自动热起动参数。

CRST/WRST 开关对自动热起动没有影响。

电源掉电后的可保持数据区

S7-300 和 S7-400 对电源掉电再上电的反映不同。

S7-300 CPU(除 CPU318 以外)只能暖起动。使用 STEP 7 可以指定位存储器、定时器、计

数器和数据块区域具有保持功能以避免由于电源掉电而丢失数据。当重新上电时，执行连同存储器在内的自动暖起动。

S7-400 对重新上电的反应依据参数设置而定，可以是暖起动(跟随保持或非保持上电)，也可以是热起动(只能是跟随可保持上电)。

下表所示是 S7-300 和 S7-400 CPU 在暖起动、冷起动或热起动过程中可保留的数据。

X	表示数据可保持
VC	表示逻辑块保留在EPROM中，所有过载的逻辑块都将丢失
VX	表示只有在EPROM中的数据块保留下来，可保持数据取自NV-RAM（在RAM中装载或生成的数据块丢失）
O	表示数据被复位或被擦除(DB块中的内容)
V	表示数据被设为来自EPROM存储器的初始值
-	由于没有NV-RAM所以不可能

下表所示在工作存储器（EPROM 和 RAM 装载存储器）中保留的数据。

EPROM（存储卡或内部集成的）									
有后备电池的CPU					没有后备电池的CPU				
数据	装载存储器中的块	工作存储器中的DB	位存储器定时器计数器	位存储器定时器计数器	装载存储器中的块	工作存储器中的DB	工作存储器中的DB	位存储器定时器计数器	位存储器定时器计数器
			（定义为可保持）	（定义为非保持）		（定义为可保持）	（定义为非保持）	（定义为可保持）	（定义为非保持）
在S7-300上暖起动	X	X	X	0	VC	VX	V	X	0
在S7-400上暖起动	X	X	X	0	VC	-	V	0	0
在S7-300上冷起动	X	0	0	0	VC	V	V	0	0
在S7-400上冷起动	X	0	0	0	VC	——	V	0	0
在S7-400上热起动	X	X	X	X		只允许暖起动			

起动的特性

下表所示为 CPU 起动过程中所作的操作：

按顺序执行的操作	暖起动	冷起动	热起动
清除I堆栈/B堆栈	X	X	0
清除非保持的位存储器、定时器、计数器	X	0	0
清除所有位存储器、定时器、计数器	0	X	0
清除过程映像输出表	X	X	可选
清除数字信号模板的输出	X	X	可选
放弃硬件中断	X	X	0
放弃延时中断	X	X	0
放弃诊断中断	X	X	X
更新系统状态列表（SZL）	X	X	X
评估模板参数并传至模板或传送缺省数值	X	X	X
执行相关的起动OB	X	X	X
执行剩余循环（由于掉电，部分用户程序没执行）	0	0	X
更新过程映像输入表	X	X	X
转换为RUN后使能数字输出（取消OD信号）	X	X	X
X 表示执行 0 表示不执行			

起动失败

如果在起动过程中出现错误，则起动失败并且 CPU 转为或保持 STOP 模式。

失败的暖起动必须再重复进行。起动失败后，暖起动和热起动都是可以执行的。

在以下情形下，起动(暖起或热起)不执行或失败：

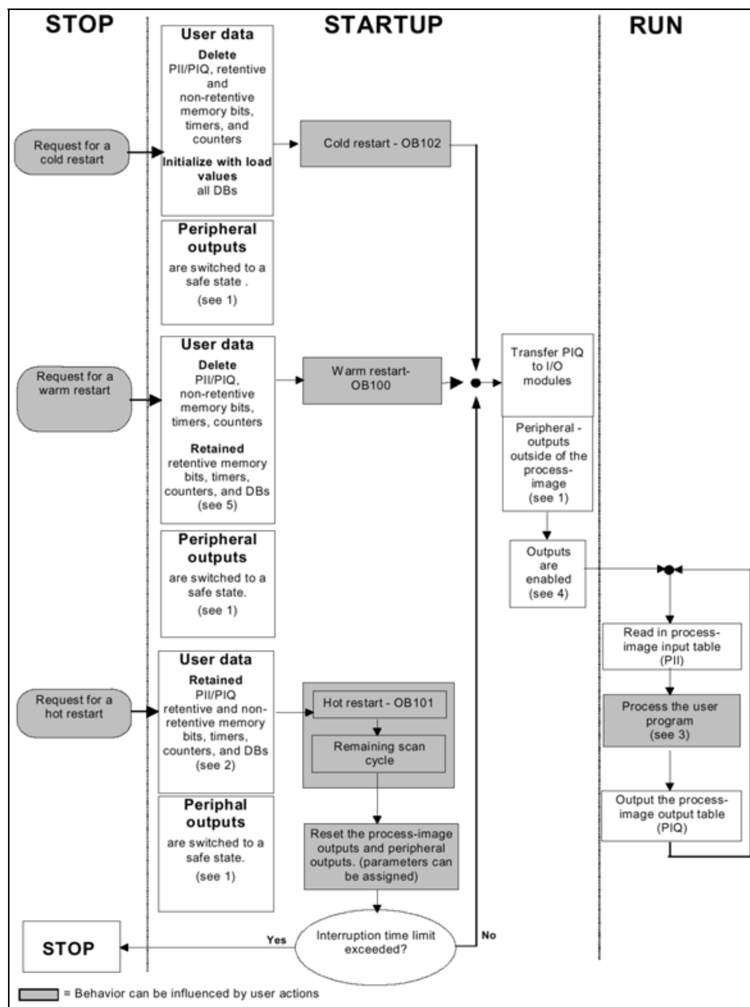
- CPU的钥匙开关设为STOP
- 请求存储器复位
- 插入的存储卡中带有非STEP 7程序(如STEP 5)
- 在单处理器模式下插入多个CPU
- CPU不能识别用户程序中的OB，或该OB已被禁用。
- 上电后，如果CPU发现用STEP 7生成的组态列表中所列的模板实际上没有全部插上(参数的预置值和实际值不同)。
- 如果评估模板参数时出错。

在以下情形下热起动不执行或被失败：

- CPU存储器被复位(存储器复位后只能暖起动)。
- 超过中断时间限制(这一时间是指包括剩余循环在内的起动OB被执行与现有RUN模式之间的时间)。
- 模板组态已改变(例如更换模板)。
- 参数赋值只允许暖起动。
- 在CPU处于STOP模式下装载、删除或修改块。

操作顺序

下图所示为 CPU 在 STARTUP 和 RUN 期间的操作：



CPU 在 STARTUP 和 RUN 期间的操作要点：

1. 通过 I/O 模板将硬件中的所有外设输出切换到安全状态(缺省值=0)。不管用户程序是否使用了过程映像区内的输出还是使用了过程映像区外的输出。

   如果使用的信号模板具有替换值功能，则可以对输出特性进行赋值，例如将输出设置为保留上次有效值。

2. 有必要处理剩余循环。

3. 当中断 OB 第一次调用后，当前的过程映像输入表对这些中断 OB 依然有效。

4. 在用户程序的首次循环中，本地外设输出以及分布式外设输出的状态通过下列步骤决定：

- 使用可进行参数赋值的输出模板，使能替换值输出或保持上次有效值。
- 对于热起动：激活 CPU 的起动参数“Reset ourputs during hot restart”，以便使其输出一个 0（对应于缺省设置）。
- 在起动 OB 中预置输出（OB100，OB101，OB102）。

5. 在没有后备的 S7-300 系统中，只能保持那些设置为保持功能的 DB 区。

A.1.4 RUN 模式

在 RUN(运行)模式，CPU 执行循环、时间驱动和中断驱动程序如下：

- 读入过程映像输入表。
- 执行用户程序。
- 输出过程映像输出表。

只有在 RUN 模式下才能使用全局数据通讯(全局数据表)，为组态连接使用通讯 SFB 以及为非组态连接使用通讯 SFC 在 CPU 之间进行主动的数据交换。

下表所示为在不同的操作模式下可能的数据交换：

通讯类型	CPU1的模式	数据交换的方向	CPU2的模式
全局数据通讯	RUN	←→	RUN
	RUN	→	STOP/HOLD
	STOP	←	RUN
	STOP	X	STOP
	HOLD	X	STOP/HOLD
单向通讯	RUN	→	RUN
调用通讯SFB	RUN	→	STOP/HOLD
调用通讯SFB的双向通讯	RUN	←→	RUN
单向通讯	RUN	→	RUN
调用通讯SFC	RUN	→	STOP/HOLD
调用通讯SFC的双向通讯	RUN	←→	RUN
←→表示　数据交换可双向进行 →表示　数据交只能单向进行 X表示　数据交换不可能			

A.1.5 HOLD 模式

HOLD(保持)模式是一种特殊模式。它只用于起动过程中或 RUN 模式下的测试目的。HOLD 模式意味着：

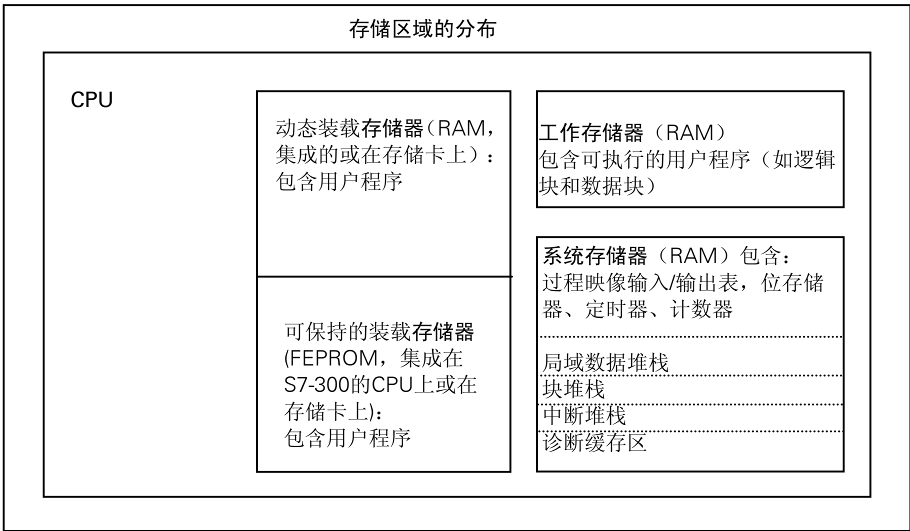
- 所有定时器被冻结：不处理定时器和CPU运行时间计数器，监控时间停止，时间驱动层的基本时钟脉冲停止。
- 实时时钟运行。
- 输出未使能，但是用于测试目的可直接使能。
- 输入和输出可被置位和复位。
- 如果在HOLD模式下，有后备电池的CPU出现掉电，当电源恢复时CPU转为STOP但不执行一个自动热起动或暖起动。没有后备电池的CPU当电源恢复时执行一个自动暖起动。
- 可以接收全局数据，可以为组态连接调用通讯SFB的被动单向通讯以及为非组态连接调用通讯SFC的被动单向通讯(见表中RUN模式)。

A.2 S7 CPU 的存储区域

A.2.1 存储区域的分布

S7 CPU 的存储器可以分为三个区域(见下图)：

- 装载存储器用于存储用户程序，不包括符号地址赋值或注释(这些保留在编程器的存储器上)。装载存储器可以是RAM或EPROM。
- 没有被标为起动所需的块只能保存在装载存储器中。
- 工作存储器(集成的RAM)包含与运行用户程序相关的那部分S7程序。程序只在工作存储器和系统存储器区域内被执行。
- 系统存储器(RAM)内包含由每个CPU为用户程序提供的存储器组件，如过程映像输入和输出表、位存储器、定时器和计数器。系统存储器还包含块堆栈和中断堆栈。
- 除了上述区域外，CPU的系统存储器还提供临时存储器(局域数据堆栈)，它包含块被调用时的临时数据。只有该块激活时这些数据才保持有效。



A.2.2 装载存储器和工作存储器

当从编程器上下载用户程序到 CPU 时，只有逻辑块和数据块能够被装载到 CPU 的装载存储器和工作存储器。

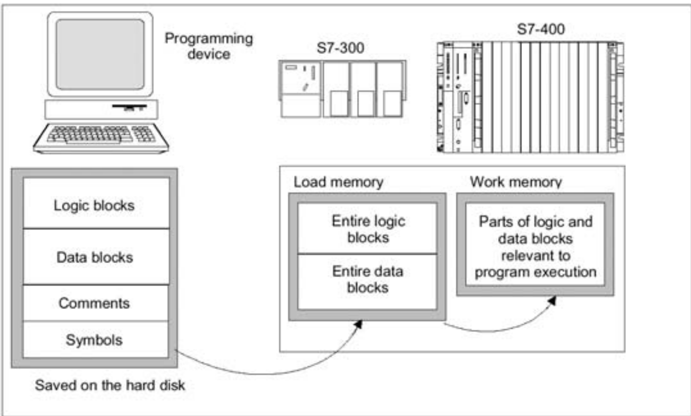
符号地址赋值(符号表)和块注释保留在编程器上。

用户程序的划分

为确保用户程序的快速执行以及对无法扩展的工作存储器减轻无必要的负担，只有与程序执行相关的块才被装载到工作存储器。

程序运行不需要的那些块(如块首)保留在装载存储器中。

下图所示为程序被装载到 CPU 的存储器中。



注意

在用户程序中使用系统功能(如 SFC22 CREAT_DB)生成的数据块全部存储在 CPU 的工作存储器中。

有些 CPU 对工作存储器中的程序和数据区分开管理。这些区域的大小和分配显示在 CPU 的模板信息的“Memory(存储器)”标签中。

将数据块标识为“与执行无关”

作为 STL 程序的一部分，用源文件编写的数据块可以被标识为“与执行无关”(关键字 UNLINKED)。这意味着当它们被下载到 CPU 时，这些 DB 块只能保存在装载存储器中。如果有必要的话，可以使用 SFC 20BLKMOV 将这些块的内容复制到工作存储器。

这种方法节省了工作存储器的空间。可扩展的装载存储器被用作缓存器(例如，一个混合剂的公式：只有用于下一批的公式被装载到工作存储器)。

装载存储器结构

装载存储器可以用存储卡进行扩展。要查找装载存储器的最大容量，可参考“S7-300 可编程控制器，硬件和安装手册”以及“S7-400，MF-400 可编程控制器模板规范参考手册”。

在 S7-300 的 CPU 中装载存储器不仅有集成的 RAM 部分还可以有集成的 EPROM 部分。数据块区域可以通过用 STEP 7 进行参数赋值声明为可保持(见 S7-300CPU 可保持存储区域)。

在 S7-400 的 CPU 上，必须使用一个存储卡(RAM 或 EPROM)来扩展装载存储器。集成的装载存储器是一个 RAM 存储器，主要用于块的重新装载和修改。新型的 S7-400 CPU，可以插入附加的工作存储器。

装载存储器在 RAM 区和 EPROM 区的性能

根据选择了 RAM 或 EPROM 存储器来扩展装载存储器，装载存储器在下载、重新装载或存储器复位过程中可能会有不同的反应。

下表所示为不同的装载方式：

存储器类型	装载方式	装载类型
RAM	装载和删除单个块	PG-CPU连接
	下载和删除一个完整的S7程序	PG-CPU连接
	重新装载单个块	PG-CPU连接
集成的（只有S7-300）或外插的EPROM	下载整个S7程序	PG-CPU连接
外插EPROM	下载整个S7程序	上传EPROM到PG以及在CPU上插入存储卡下载EPROM到CPU

当复位 CPU 存储器(MRES)或者移走 CPU 或 RAM 存储卡时，会丢失存储在 RAM 中的程序。

存储在 EPROM 存储卡上的程序不会被 CUP 存储器复位擦除，即使没有后备电池数据也能保持(传送、备份拷贝)。

A.2.3 系统存储器

A.2.3.1 使用系统存储器区域

S7 CPU 的系统存储器被划分为地址区域(见下表)。在用户程序中使用指令可以在相应的地址区内直接对数据进行寻址。

地址区域	通过以下单位访问	S7记号（IEC）	描述
过程映像输入表	输入（位）	I	在扫描循环开始处，CPU读取输入模板的输入并在该区域内记录数值。
	输入字节	IB	
	输入字	IW	
	输入双字	ID	
过程映像输出表	输出（位）	Q	在扫描循环过程中，程序计算输出值并将它们放在该区域。在扫描循环结束处，CPU将这些计算输出值发送到输出模板。
	输出字节	QB	
	输出字	QW	
	输出双字	QD	
位存储器	存储（位）	M	该区域为用户程序中的中间计算结果提供存储。
	存储字节	MB	
	存储字	MW	
	存储双字	MD	
定时器	定时器（T）	T	该区域为定时器提供存储。
计数器	Counter(C)	C	该区域为计数器提供存储。
数据块	数据块，用"OPN DB"打开	DB	数据块中包含程序需要的信息。它们可以被定义为由所有逻辑块通用（共享DB）或分配给某个特定的FB或SFB（背景DB）
	数据位	DBX	
	数据字节	DBB	
	数据字	DBW	
	数据双字	DBD	
	数据块，用"OPN DI"打开	DI	
	数据位	DIX	
	数据字节	DIB	
	数据字	DIW	
	数据双字	DID	
局域数据	局域数据值	L	该区域包含块被执行时的临时数据。L堆栈还为传送块参数以及记录来自梯形逻辑段的中间结果提供存储器。
	局域数据字节	LB	
	局域数据字	LW	
	局域数据双字	LD	
外设（I/O）区域：输入	外设输入字节	PIB	外设输入和输出区域允许直接访问中央的和分布式的输入和输出模板（DP）。
	外设输入字	PIW	
	外设输入双字	PID	
外设（I/O）区域：输出	外设输出字节	PQB	
	外设输出字	PQW	
	外设输出双字	PQD	

参考以下 CPU 手册或指令集，查找 CPU 上能够使用的地址区域的信息：

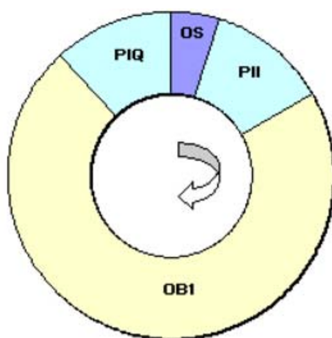
- “S7-300可编程控制器，硬件和安装”手册
- “S7-400，M7-400可编程控制器，模板规范”参考手册。
- “S7-300可编程控制器，指令列表”
- “S7-400可编程控制器，参考指南”

A.2.3.2 过程映像输入/输出表

在用户程序中访问输入(I)和输出(Q)地址区时，程序并不直接访问数字信号模板上的信号状态而是访问 CPU 的系统存储器中的存储区和分布式 I/O。这一存储区域就是过程映像区。

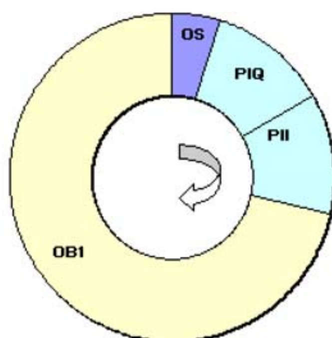
更新过程映像

下图所示为 10/98 以前 CPU 的一个扫描循环周期内过程处理的步骤：



操作系统的内部任务之一是将输入状态读入到过程映像输入表(PII)，一旦该步完成，用户程序将执行所调用的所有块。循环结束时，将过程映像输出表(PIQ)写入模板的输出区。操作系统将单独控制过程映像输入表的读入和过程映像输出表的输出。

下图所示为 10/98 以后的 CPU 的一个扫描循环周期内过程处理的步骤：



操作系统的内部任务之一是将过程映像输出表(PIQ)写入模板的输出区，并将输入状态读入到过程映像输入表(PII)，一旦该步完成，用户程序将执行所调用的所有块。操作系统将单独控制过程映像输入表的读入和过程映像输出表的输出。

过程映像区的优点

与直接访问输入/输出模板相比，访问过程映像的优点在于：在一个程序循环中 CPU 有一个一致的过程信号映像。如果在程序执行过程中输入模板上的信号状态变化，过程映像中的信号状态保持不变，直到下一个循环过程映像才再次更新。

由于过程映像区位于 CPU 内部，访问过程映像所需的时间比直接访问信号模板要少得多。

更新过程映像区的部分区域

除了操作系统对过程映像输入表(PII)和过程映像输出表(PIQ)自动更新外，S7-400 CPU 可以最多对 15 个过程映像的输入区域进行参数赋值。（见“S7-400 可编程控制器，硬件和安装手册”以及“S7-400，M7-400 可编程控制器模板技术规范参考手册”）。这意味着当需要时，可以更新过程映像表的部分区域，而不依赖过程象表的循环更新。

STEP 7 对过程映像区分配的每个输入/输出地址不再属于 OB1 过程映像输入/输出表。只能用 OB1 过程映像区和所有的过程映像区对输入/输出地址分配一次。

使用 STEP 7 分配地址时可以定义过程映像区分区(在过程映像区中列出了模板的输入/输出地址)。可以通过用户程序，也可以用 SFC 更新过程映像区。

例外：系统不能在同步循环中断 OB 中更新过程映像区分区，即使它们连接到 OB61 至 OB64。

提示

在 S7-300 CPU 上，未使用的输入和输出过程映像表可用作附加的位存储区。使用这一选项的程序则不能运行在 S7-400 的 CPU 上(4/99 以前的 CPU)。

对于 S7-400 CPU

- 作为位存储区的过程映像区不能位于所赋值的“过程映像区大小”内。
- 或者不能位于系统或SFC26/27所能更新的过程映像区内。

使用 SFC 更新过程映像区

通过使用下列 SFC，用户程序可以更新整个过程映像区或过程映像区分区：

- 要求：不能更新有问题的过程映像。
- SFC26 UPDAT_PI：更新过程映像输入表。
- SFC27 UPDAT_PO：更新过程映像输出表。

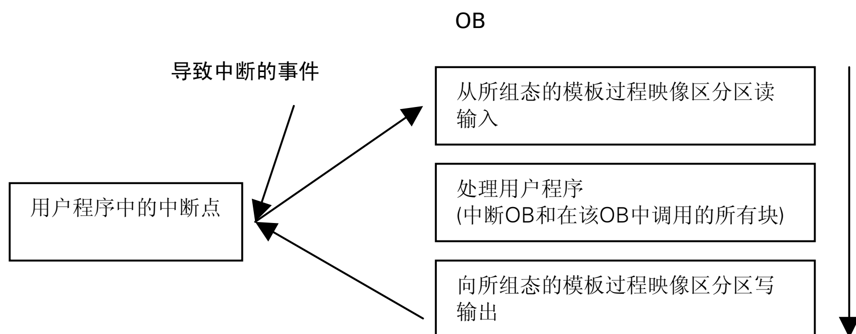
过程映像区分区的系统更新

与在 OB1 处理前或处理后周期更新整个过程映像相似，可以通过调用一个 OB 让系统自动更新过程映像区分区。对于一些特殊的 CPU，可以将该功能作为参数进行赋值。

在运行期间，可以自动地更新所分配的过程映像区分区：

- 在OB处理前，过程映像区分输入
- 在OB处理后，过程映像区分区输出

可以对 CPU 连同 OB 的优先级进行参数赋值，用来指示哪个过程映像区分区分配给哪个 OB。



更新过程映像期间发生 I/O 访问错误(PZF)

在更新过程映像期间，CPU 家族(S7-300 和 400)对一个错误的缺省响应有以下不同：

- S7-300：诊断缓冲区没有相应条目，相应的输入字节复位为0并在故障排除前始终为0。
- S7-400：诊断缓冲区有相应的条目，启动OB85对每个相应的过程映像进行I/O访问。每次访问过程映像时对有故障的输入字节复位为0。

对于 4/99 以后的 CPU，可以对 I/O 访问错误的响应重新进行参数赋值：

- 生成一个诊断缓冲区的条目，并对I/O访问错误启动OB85(在调用OB85前，有故障的输入字节复位为0，在故障排除前系统不会对该字节进行改写)
- 产生S7-300的默认响应(不调用OB85，有故障的输入字节复位为0，在故障排除前系统不会对该字节进行改写)
- 产生S7-400的默认响应(每次访问均调用OB85,对过程映像访问时对有故障的输入字节复位为0)

OB85 的起动频率

除了对已进行参数赋值（到来/离去，或每次 I/O 访问）的 PZF 进行响应外，一个模板的地址空间同时也决定了 OB85 的起动频率：

对于一个地址空间最多为双字的模板，OB85 起动一次。例如：对于一个最大为 32 个输入或输出的数字量模板，或一个 2 通道的模拟量模板。

对于有大量地址空间的模板，根据双字命令访问的频率起动 OB85。例如 4 通道模拟量模板起动 2 次 OB85。

A.2.3.3 局域数据堆栈

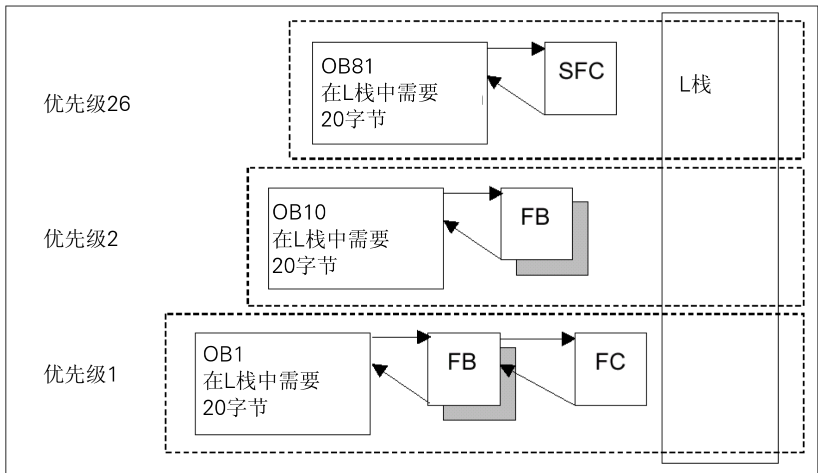
L 堆栈存储以下内容：

- 块的局域数据的临时变量
- 组织块的起动信息
- 传递参数的信息
- 在梯形逻辑程序中的逻辑中间结果

当编写组织块时，可以声明临时变量(TEMP)，这些临时变量只在该块执行时是有效的，然后就被覆盖了。在首次访问局域数据堆栈前，这些局域数据必须被初始化。除此之外，每个组织块还需要 20 个字节的局域数据存储其起动信息。

CPU 对当前正在执行的块的临时变量(局域数据)所用的存储器数量是有限制的。这个存储区域的大小、局域数据堆栈的大小要依据 CPU 而定。局域数据堆栈在各优先级中平等划分(缺省设置)。这意味着每个优先级有自己的局域数据区域，因此，保证了高优先级及其 OB 也有用于它们的局域数据的空间。

下图所示，在一个 OB1 被 OB10 中断，OB10 又被 OB81 中断的例子中说明不同优先级在 L 堆栈中局域数据的分配。





注意

OB 及其相关块的所有临时变量(TEMP)都存储在 L 堆栈。如果在执行块时有太多层嵌套，L 堆栈可能会溢出。

如果超过了一个程序所允许使用的 L 堆栈的大小，S7 的 CPU 转为 STOP 模式。

测试程序中的 L 堆栈(临时变量)。

同步错误 OB 所需的局域数据必须考虑在内。

为优先级分配局域数据

不是每个优先级都需要同等数量的局域数据堆栈的存储区域。通过在 STEP 7 中进行参数赋值，可以为 S7-400 CPU 和 CPU318 的各个优先级分配不同大小的局域数据区域。不需要的优先级可以不选择，对 S7-400 CPU 和 CUP318 来说，其它优先级的存储区域则增加了。未激活的 OB 在程序执行过程中被忽略，从而节省循环时间。

对于其它的 S7-300 CPU，每个优先级分配有固定数量的局域数据(256 字节)，不能够改变。

A.2.3.4 中断堆栈

如果程序的执行被更高优先级的 OB 中断，操作系统将当前累加器和地址寄存器中的内容以及打开的数据块的号码和大小保存在中断堆栈中。

一旦新 OB 执行完了，操作系统从中断堆栈装载信息，从被中断的块的断点处继续执行。

当 CPU 在 STOP 模式时，可以在编程器上用 STEP 7 显示 I 堆栈。可以发现 CPU 转为 STOP 模式的原因。

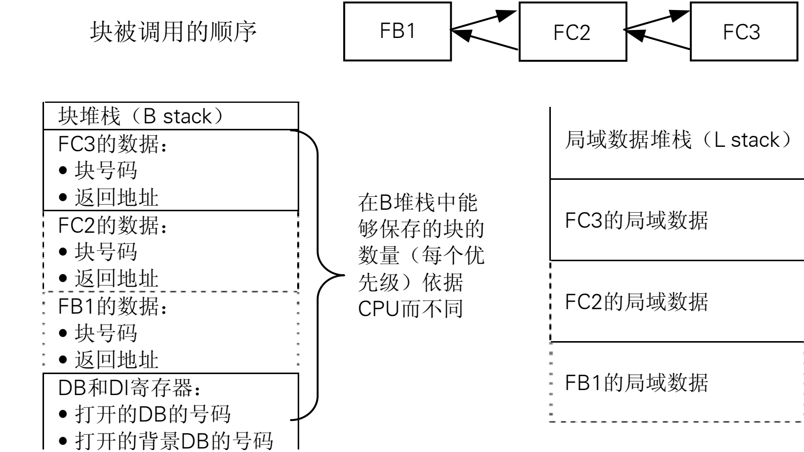
A.2.3.5 块堆栈

如果一个块的处理被另外一个块的调用或更高的优先级(中断/故障服务)中断，B 堆栈存储以下数据：

- 被中断块的号码、类型(OB、FB、FC、SFB、SFC)和返回地址。
- 在被中断块中打开的数据块的号码(从DB和DI寄存器)。

利用这些数据，用户程序在被中断后则可继续执行。

如果 CPU 在 STOP 模式，可以在编程器上用 STEP 7 显示 B 堆栈。B 堆栈中列出所有在 CPU 转为 STOP 模式时已被调用但还未完全执行的块。这些块按照处理启动的顺序排列(见下图)。



数据块寄存器

有两个数据块寄存器。它们包含打开的数据块的号码：

- DB寄存器中包含打开的共享数据块的号码
- DI寄存器中包含打开的背景数据块的号码

A.2.3.6 诊断缓冲器

诊断缓冲器按照其发生的顺序显示诊断消息。第一条是最新发生的事件。诊断缓冲器中条目的数量取决于所使用的模板及当前的运行模式。

诊断事件包括：

- 一个模板的故障
- 接线错误
- CPU中的系统故障
- CPU的模式转换
- 用户程序中的错误
- 用户定义的诊断事件(通过系统功能SFC52实现)

A.2.3.7 评估诊断缓存器

系统状态列表的一部分就是诊断缓冲区，这里包括更多的关于系统诊断事件和用户定义的诊断事件的信息，并按其出现的顺序排列。当系统诊断事件出现时写入诊断缓冲区的信息与传送至相应组织块的启动信息是一致的。

无法清除诊断缓存区中的各项条目及其内容，即使存储器全清之后它们也会保留下来。

诊断缓存区提供以下可能：

- 如果CPU转为STOP模式，可以评估最后的引起停机的事件并确定起因的位置。
- 能够更快地检测故障的原因，增加了系统的有效性。
- 可以评估和优化动态的系统响应。

组织诊断缓存区

诊断缓冲区被设计用作一个环式缓冲器，其最多输入项的数量依各模板而定。这意味着当达到最大输入项数时，下一个诊断缓冲事件使最早的一项被删除。所有输入项向后移一个置位。即最新的输入总是诊断缓冲区的第一项。对于 S7-300 CPU314，可能的输入项数是 100：

显示的诊断缓存区中输入条目依模板及其当前操作模式而定。有些 CPU 可以设置其诊断缓存区的长度。

诊断缓存内容

诊断缓存中上部列表框中包含所有出现的诊断事件并有以下信息：

- 输入条目的序列号(最新的输入项是1号)
- 诊断事件的时间和日期：如果模板有集成的时钟则显示该模板的时间和日期。保持缓存区中的时间数据有效，为模板设置时间和日期并定期检查是非常重要的。
- 对诊断事件的简短描述。

在诊断缓存下部的文本框中，显示在上部窗口列表中选中事件的所有附加信息。这些信息包括：

- 事件号
- 事件描述
- 由诊断事件引起的模式转换
- 引起该条目进入缓存区的错误在块中所处位置的索引(块类型、块号码、相对地址)
- 到来或离去的事件状态
- 事件特定的附加信息

用“Help on Event(事件帮助)”按钮可以显示在上部列表框中选中事件的附加信息。

在系统块和系统功能中的参考帮助中可以得到事件 ID 的信息。

保存内容到一个文本文件

在“Module Information(模板信息)”对话框的“Diagnostic Buffer(诊断缓存区)”标签中使用“Save As(另存为)”按钮，可以将诊断缓存区中的内容存为 ASCII 文本。

显示诊断缓存区

通过“Module Information”对话框中的“Diagnostic Buffer”标签，或在程序中使用系统功能 SFC5l RDSYSST，可以在编程器上显示诊断缓存区的内容。

停机前的最后输入项

可以指定将 RUN 转为 STOP 前最后一个诊断缓存条目送到一个登录过的监视设备上(如 PG、OP、TD)，以便快速锁定和纠正引起 STOP 模式的错误。

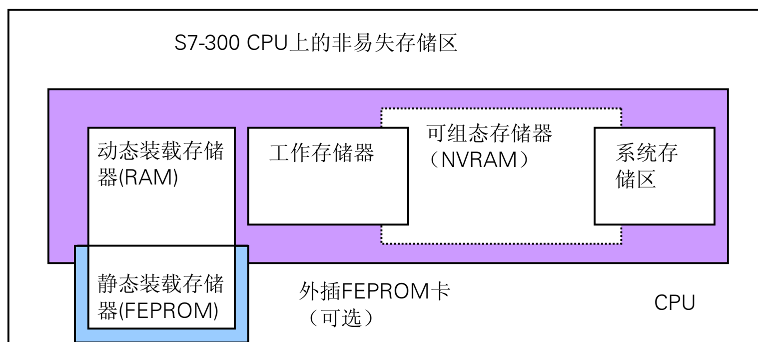
A.2.3.8 S7-300 CPU 上的保持存储区

如果出现电源掉电或 CPU 存储器复位(MRES)，S7-300 的存储器(动态装载存储器(RAM)、工作存储器和系统存储器)被复位并且以前保存在这些区域的所有数据全部丢失。对 S7-300 CPU，可以用以下方法保护程序及其数据：

- 使用后备电池保护存在装载存储器、工作存储器以及部分系统存储器中的全部数据。
- 可以将程序存在EPROM(既可以是存储卡也可以是集成在CPU上的，参考“S7-300可编程控制器，硬件和安装”手册)。
- 依据CPU的不同，可以存储一定数量的数据在非易失区域NVRAM。

使用 NVRAM

S7-300 CPU 在 NVRAM(非易失 RAM)中提供一个区域(见下图)。如果已将程序存储到装载存储的 EPROM 中，可以通过组态 CPU 保存一定的数据(如果出现掉电或当 CPU 从 STOP 转为 RUN)。



要实现这一功能，设置 CPU 使以下数据保存在非易失 RAM 中：

- DB中包含的数据(如果已将程序保存在装载存储器的EPROM中，这样做才有用)
- 定时器和计数器的值
- 位存储的数据

在每个 CPU 上，可以存储一定数量的定时器、计数器和存储位，还可以指定一定数量的字节用以存储 DB 中包含的数据。

CPU 的 MPI 地址存储在 NVRAM 中。这就确保了 CPU 在掉电或存储器复位后可以进行通讯。

使用后备电池保护数据

使用后备电池，装载存储器和工作存储器在掉电期间是可以保持的：如果组态 CPU，使定时器、计数器和位存储器保存在 NVRAM 中，则不论是否使用了后备电池，这些信息都可以保持。

组态 NVRAM 的数据

当用 STEP 7 组态 CPU 时，可以决定哪个存储区域是可保持的。可以被组态到 NVRAM 中的存储器的数量根据所使用的 CPU 而定。所备份数量不能比 CPU 所指定的数目多。

A.2.3.9 S7-400 CPU 上的可保持存储区域

无后备电池的操作

如果在没有后备电池的情况下操作系统，当电源掉电出现时或复位 CPU 存储器(MRES)时，S7-400 CPU 的存储器(动态装载存储器(RAM)、工作存储器和系统存储器)被复位，这些区域中所包含的数据全部丢失。

没有后备电池，只能作暖起动并且没有可保持区域。电源掉电后，只有 MPI 参数(如 CPU 的 MPI 地址)被保留下来。这意味着在电源掉电或存储器复位后仍保持有通讯的能力。

有后备电池的操作

如果使用电池备份存储器：

- 电源掉电后，当CPU再起动时，所有RAM区域的全部内容都被保留。
- 在暖起动过程中，位存储器、定时器和计数器的地址区被清除。数据块的内容被保留。
- 除了被设定为非保持的位存储器、定时器和计数器之外，RAM工作存储器中的内容也都保留。

组态保持数据区域

可以声明一定数量的存储位、定时器和计数器具有保持功能(数量根据所使用的 CPU 而定)。在暖起动过程中，当使用后备电池时，这些数据也可保留。

当用 STEP 7 参数赋值时，可以定义哪些位存储器、定时器和计数器在暖起动时要保留。只能备份 CPU 所允许的数量的数据。

更多的关于定义可保持存储器区域的信息，可参考“S7-400，M7-400 可编程控制器，模板技术规范”参考手册。

A.2.3.10 在工作存储器中的可组态存储器对象

有些 CPU 可以在 HWConfig(硬件组态)中设置诸如局域或诊断缓存区对象的大小。例如，缺省值，在其它地方的工作存储器就可以获得一个较大的部分。这些 CPU 的设置可以在 CPU 模块属性“Memory(存储器)”标签中显示(“Details(详细数据)”按钮)。

在修改存储器组态并下载到可编程控制器以后，为使修改生效，必须执行一个冷起动。

A.3 数据类型和参数类型

A.3.1 介绍数据类型和参数类型

用户程序中的所有数据必须标有一个数据类型。可用的数据类型有以下几种：

- STEP 7提供的基本数据类型
- 通过组合基本数据类型而生成的复合数据类型
- 用来定义传送给FB或FC参数的参数类型

一般信息

语句表、梯形逻辑和功能块图指令使用特定大小的数据对象。位逻辑指令使用位。例如，装载和传送指令(STL)以及 move(移动)指令(LAD 和 FBD)使用字节、字和双字。

一个位是一个二进制数字“0”或“1”。一个字节由八个位组成，一个字十六位，一个双字三十二位。

数学指令也使用字节、字或双字。在这些字节、字或双字的地址中可以用各种格式编码，比如整数和浮点数。

当使用符号寻址时，定义符号并且要为这些符号指定一个数据类型(见下表)。不同的数据类型有不同的格式选项和数字表示法。

本章只描述了数字和常数的若干写法。下表列出一些数字和常数的格式不再作详细的解释。

格式	所占位的多少	数值表示法
Hexadecimal	8,16 和 32	B#16#,W#16#,and DW#16#
Binary	8,16 和 32	2#
IEC date	16	D#
IEC time	32	T#
Time of day	32	TOD#
Character	8	‘A’

A.3.2 基本数据类型

每个基本数据类型有个规定的长度。下表列出了基本数据类型。

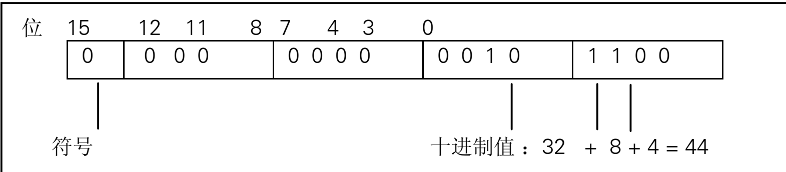
类型和描述	所占位数	格式选项	范围及数值表示法（最低值至最高值）	示例
BOOL(位)	1	布尔文本	TRUE/FALSE	TRUE
BYTE (字节)	8	十六进制数	B 16#0 to B 16#FF	L B#16#10 L byte#16#10
WORD (字)	16	二进制数 十六进制数 BCD 无符号的十进制数	2#0 to 2#1111_1111_1111_1111 W#16#0 to W#16#FFFF C#0 to C#999 B#(0,0)to B#(255,255)	L 2#0001_0000_0000_0000 L W#16#1000 L word16#1000 L C#998 L B#(10,20) L byte(10,20)
DWORD (双字)	32	二进制数 十六进制数 BCD 无符号的十进制数	2#0 to 2#1111_1111_1111_1111 1111_1111_1111_1111 DW#16#0000_0000 to DW#16#FFFF_FFFF B#(0,0,0,0)to B#(255,255,255,255)	2#1000_0001_0001_1000_ 1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1,14,100,120) L byte#(1,14,100,120)
INT (整数)	16	有符号的十进制数	-32768 to 32767	L 1
DINT (整数，32位)	32	有符号的十进制数	L#-2147483648 to L#2147483647	L L#1
REAL (浮点数)	32	IEEE 浮点数	上限:±3.402823e+38 下限:±1.175495e-38	L 1.234567e+13
S5TIME (SIMATIC 时间)	16	S7 时间，每步10ms (缺省值)	S5T#0H_0M_0S_10MS to S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#0H_1H_1M_0S_0MS
TIME (IEC时间)	32	IEC时间，每步1ms，带符号整数	-T#24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (IEC日期)	16	IEC 日期，每步 1天	D#1990-1-1 to D#2168-12-31	L D#1996-3-15 L DATE#1996-3-15
TIME_OF_DAY (时间)	32	时间每步 1 ms	TOD#0:0:0.0 to TOD#23:59:59.999	L TOD#1:10:3.3 L TIME_OF_DAY#1:10:3.3
CHAR (字符)	8	ASCII 字符	'A','B'etc.	L 'E'

A.3.2.1 数据类型 INT 的格式(16 位整数)

一个整数有一个符号指示它是一个正整数还是一个负整数。一个整数(16 位)在存储器中所占空间为一个字。下表所示为一个整数的范围(16 位)。

格式	范围
整数(16位)	-32768至+32767

下图为整数+44 的二进制码的表示。

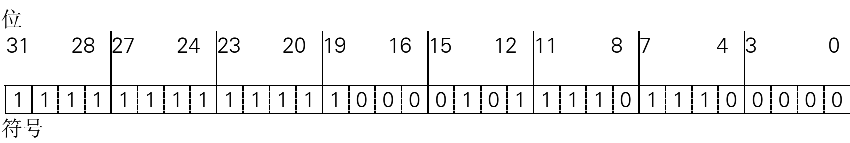


A.3.2.2 数据类型 DINT 的格式(32 位整数)

一个整数有一个符号指示它是一个正整数还是一个负整数。一个双整数占用的存储空间为两个字，下表所示为一个双整数的范围。

格式	范围
整数(32位)	-2.147,483,648至+2,147,483,647

下图所示为整数-500,000 的二进制码的表示。在二进制系统中，整数的负值形式表示为这个整数的二维补码。可以通过对一个整数的各位取反然后加 1 得到二维补码。



A.3.2.3 数据类型 REAL 的格式(浮点数)

浮点数格式的表达通式为“数值=m*E 的 b 次幂”。基数“b”和指数“E”是整数；尾数“m”是有理数。

这种数值表达方式的优点是在有限的空间内即可以表示一个非常大的数也可以表示一个非常小的数。使用有限位数的尾数和指数，可以覆盖一个较宽范围内的数字。

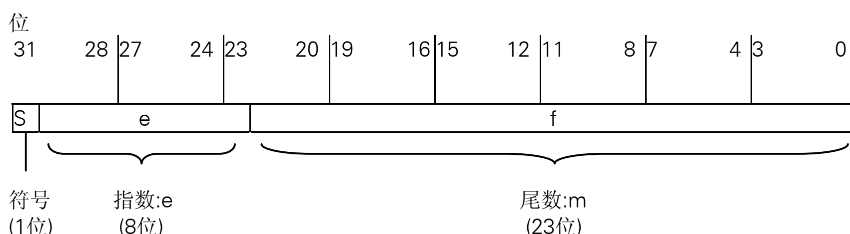
缺点则在于对计算的精确度的限制。例如，当求两个数的和时，必须通过移动尾数(因此浮动小数点)使指数对阶，因为只有具有相同指数的数值可以相加。

STEP 7 中的浮点数格式

STEP 7 中的浮点数符合二进制浮点运算 IEEE 标准中 ANSI/IEEE 标准 754-1985 有关基本格式、单宽的描述。它们包括以下部分：

- 符号S
- 指数e=E+bias(偏移)，通过一个常数(偏移=+127) 增加
- 尾数m的小数部分。尾数的整数部分不和余数存在一起，因为在有效数值范围内它总是等于1。

这三个部分一共占用一个双字(32 位)：



下图所示为一个浮点数格式各个位数值。

浮点数的组成元素	位号	数值
符号S	31	
指数e	30	2的7次幂
…	…	…
指数e	24	2的1次幂
指数e	23	2的0次幂
尾数m	22	2的-1次幂
…	…	…
尾数m	1	2的-22次幂
尾数m	0	2的-23次幂

使用三个组成部分 s，e 和 m，这种形式的数值可以由以下公式定义：

数值=1.m*2 的(e bias)次幂，这里：

- e：1≤e≤254
- Bias：bias=127。这意味着指数不再另外需要一个符号。
- S：正数S=0，负数S=1。

浮点数的数值范围

使用以上所示的浮点数格式，有以下结果：

- 最小的浮点数=1.0*2的(1-127)次幂=1.0*2的(-126)次幂=1.175495E-38
- 最大的浮点数=2-2的(-23)次幂*2的(254-127)次幂=2-2的(-23)次幂*2的(+127)次幂=3.402823E+38

零值表示为 e=m=0；e=255 和 m=0 表示“infinite(无穷大)”。

格式	范围[1)]
符合ANSI/IEEE标准的浮点数	-3.402 823E+38至-1.175 495E-38和0以及 +1.175 495E-38至3.402 823E+38

下图所示为使用了非有效范围内的浮点数的指令结果的状态字中各位的信号状态：

无效的结果范围	CC1	CC0	OV	OS
-1.175494E-38<结果<-1.401298E-45(负数)下溢	0	0	1	1
+1.40298E-45<结果<+1.175494E-38(正数)下溢	0	0	1	1
结果<-3.402823E+38(负值)上溢	0	1	1	1
结果>3.402823E+38(正值)上溢	1	0	1	1
不是有效的浮点数或者无效指令（输入值在有效范围之外）	1	1	1	1

使用数字操作时请注意：

例如，当试图对-2 求平方根时，会得到“非有效的浮点数”的结果。所以，在开始基于某个结果的基础上要继续计算前，总是要首先评估状态位。

修改变量时请注意：

例如，将一个浮点数数值存储在一个存储双字时，可以以位的形式对这些数作修改，可是，并非所有的位模式都是一个有效数字。

浮点数计算的精确度



注意

涉及包含非常大和非常小的一长串数值的计算时，可能会产生不精确的结果

在 STEP 7 中浮点数可以精确到 6 个十进制位。因此，当你输入浮点数常数时，最多只能指定 6 位。

提示

计算的精确度为 6 位意味着，例如，如果数值 1 大于数值 2*10 的 y 次幂，这里 y>6 则数值 1+数值 2=数值 1

100 000 000+1=100 000 000

浮点数格式的数字示例

下图所示为以十进制值值表示的浮点数格式：

- 10.0
- P(3.141593)
- 2的平方根(P2=1.414214)

第一个示例中的数值 10.0 的浮点数格式(十六进制表示：4120 0000)如下：

e=2 的 1 次幂+2 的 7 次幂=2+128=130

m=2 的(-2)次幂=0.25

这样就可得到：

(1+m)*2 的(e-bias)次幂=1.25*2 的(130-127)次幂=1.25×2 的 3 次幂=10.0

Decimal value 10.0

Hexadecimal value	4	1	2	0	0	0	0	0
Bits	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
	0 1 0 0	0 0 0 1	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

Sign of Mantissa: s (1 bit)

Exponent: e (8 bits)

Mantissa: m (23 bits)

e = 27 + 21 = 130
1.f _ 2e-bias = 1.25 _ 23 = 10.0
[1.25 _ 2(130-127) = 1.25 _ 23 = 10.0]

f = 2-2 = 0.25

Decimal value 3.141593

Hexadecimal value	4	0	4	9	0	F	D	C
Bits	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
	0 1 0 0	0 0 0 0	0 1 0 0	1 0 0 1	0 0 0 0	1 1 1 1	1 1 0 1	1 1 0 0

Sign of Mantissa: s (1 bit)

Exponent: e (8 bits)

Mantissa: m (23 bits)

Decimal value 1.414214

Hexadecimal value	3	F	B	5	0	4	F	7
Bits	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
	0 0 1 1	1 1 1 1	1 0 1 1	0 1 0 1	0 0 0 0	0 1 0 0	1 1 1 1	0 1 1 1

Sign of Mantissa: s (1 bit)

Exponent: e (8 bits)

Mantissa: m (23 bits)

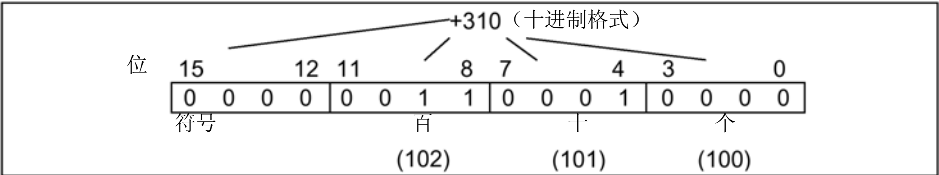
A.3.2.4 数据类型为字和双字的二进制编码的十进制数的格式

二进制编码的十进制数(BCD)格式是用一组二进制数(位)来表达一个十进制数。每 4 个位一组表示一个带符号的十进制数或十进制数的符号。几个 4 位组形成一个字(16 位)或双字(32 位)。最高有效位的 4 位指示数值和符号(1111 指示减号，0000 指示加号)使用 BCD 编码地址的命令只评估最高位(字的 15 位，双字的 31 位)。下表所示为这两种 BCD 码的格式和范围。

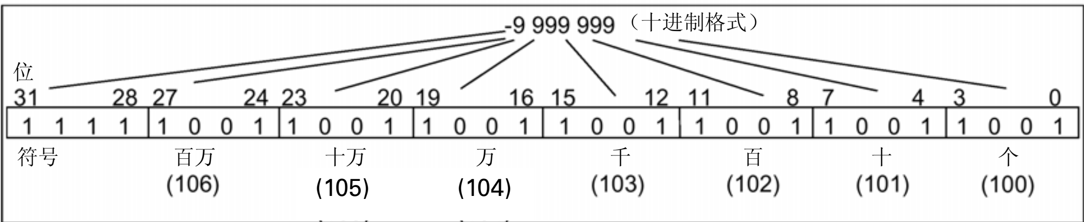
格式	范围
字(Word) (16位，3位BCD码及符号)	-999至+999
双字(Double word) (32位，7位BCD码及符号	-9 999 999至+9 999 999

下图是二进制编码的十进制数的示例：

- 字格式

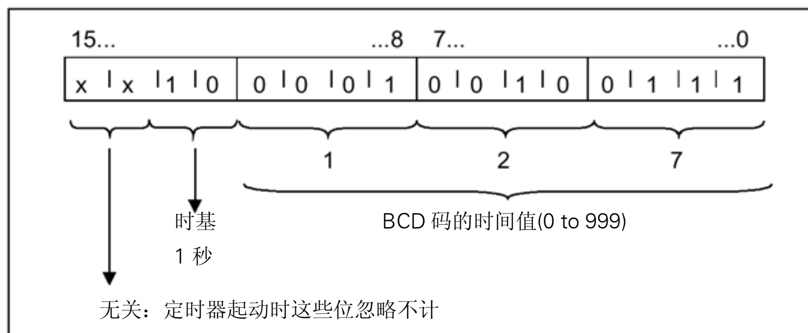


- 双字格式



A.3.2.5 数据类型 S5TIME(时间长度)的格式

当使用 S5TIME 数据类型输入时间长度时，输入以二进制编码的十进制格式存储。下图所示的时间值为 127，时基为 1s 的时间地址中的内容。



当使用 S5TIME 时，要输入一个 0-999 范围内的时间值并指明时基(见下表)。时基指示定时器以什么样的间隔一个单位一个单位地递减时间值，直至达到 0。

S5TIME 的时基

时基	时基的二进制码
10 ms	00
100 ms	01
1 s	10
10 s	11

可以使用下列语法格式预装一个时间值：

- L[1)] W#16#wxyz
  - 这里 w=时基(即时间间隔或分辨率)
  - 这里 xyz=二进制编码的十进制格式的时间值
- L[1)] S5T#aH_bbM_ccS_dddMS
  - 这里 a=小时，bb=分钟，cc=秒，dd=毫秒
  - 时基自动选择，时间值按其所取时基取整为下一个较小的数。

可以输入的最大时间值是 9, 990 秒，或 2H_46M_30S。

[1)] =L 只能用 STL 编写。

A.3.3 复合数据类型

复合数据类型可以定义大于 32 位的数据记录或包括其它数据类型的数据记录。STEP 7 允许以下复合数据类型：

- DATE_AND_TIME（日期和时间)
- STRING（字符串)
- ARRAY（数组)
- STRUCT（结构体)
- UDT（用户定义的数据类型)
- FB和SFB

下表描述了复合数据类型。可以在逻辑块的变量声明中或数据块中定义结构体和数组。

数据类型	描述
DATE_AND_TIME DT	定义一个64位(8个字节)的区域。这种数据类型以二进制编码的十进制格式存储：
STRING	缺省定义一个最多可有254个字符（数据类型CHAR）的组。存储字符串的标准区域长256字节。这是存储254个字符及2字节首部所需要的空间，可以定义需要存储在该字符串中的字符的数量来减少一个字符串所需求的存储器。（如：string[9]'Siemens”）
ARRAY	定义同一数据类型（基本的或复合的）的多维组。例如：”ARRAY[1..2,1..3]OF INT”定义了一个2×3格式的整数数组。可以使用下标（”[2,2]”）访问存储在一个数组中的数据。可以在一个数组中最多定义6维。下标可以是任意整数(-32768到+32767)
STRUCT	定义一组任意的数据类型的组合。例如，可以定义一个由结构体组成的数组或一个由结构体和数组组成的结构体
UDT	当生成数据块或在变量声明中声明变量时，可以简化大量数据的构造和数据类型的输入。在STEP 7中，可以组合复合和基本数据类型以生成“用户定义的” 数据类型。UDT有它们自己的名字并可以多次使用
FB,SFB	决定背景数据块的结构并允许在一个背景DB中传送几个FB调用的背景数据

结构体数据类型存储时与字地址的限制一致(WORD 排列)。

A.3.3.1 数据类型 DATE_AND_TIME 的格式

当使用 DATE_AND_TIME 数据类型(DT)输入日期和时间时，输入以二进制编码的十进制格式存在 8 个字节中。DATE_AND_TIME 数据类型范围如下：

DT#1990-1-1-0:0:0.0 至 DT#2089-12-31-23:59:59.999

下列所示为 1993 年 12 月 25 日星期四上午 8 点 01 分 1.23 秒的日期和时间语法，可有以下两种格式：

- DATE_AND_TIME#1993-12-25-8:01:1.23
- DT#1993-12-25-8:01:1.23

以下特殊的 IEC(国际电工技术委员会)标准功能可以对 DATE_AND_TIME 数据类型数据进行槽：

- 将某一天的日期和时间转换为DATE_AND_TIME数据格式： FC3：D_TOD_DT
- 从DATE_AND_TIME格式中提取日期： FC6：DT_DATE
- 从DATE_AND_TIME格式中提取星期： FC7：DT_DAY
- 从DATE_AND_TIME格式中提取日时钟： FC8：DT_TOD

下表所示为包含日期和时间格式字节中的内容，该日期和时间包含以下信息：星期四，12 月 25 日，1993，早晨 8:01 又 1.23 秒。

字节	内容	举例
0	年	B#16#93
1	月	B#16#12
2	日	B#16#25
3	小时	B#16#08
4	分	B#16#01
5	秒	B#16#01
6	毫秒的高两位	B#16#23
7 (4MSB)	毫秒的低两位	B#16#0
7 (4LSB)	星期 1=周日 2=周一 … 7=周六	B#16#5

数据类型 DATE_AND_TIME 允许的范围是：

- 最小：DT#1990-1-1-0:0:0.0
- 最大：DT#2089-12-31-23:59:59.999

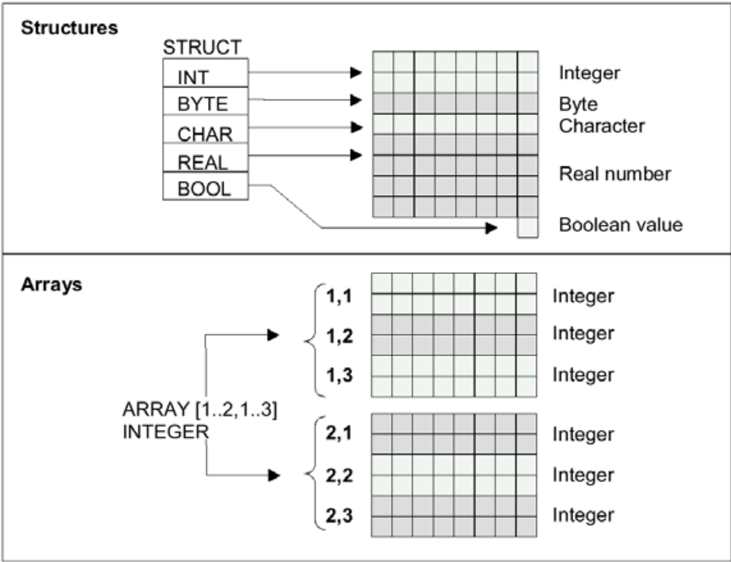
	可能的数据范围	BCD码
年	1990-1999 2000-2089	90h-99h 00h-89h
月	1-12	01h-12h
日	1-31	01h-31h
小时	00-23	00h-23h
分	00-59	00h-59h
秒	00-59	00h-59h
毫秒	00-999	000h-999h
星期	Sunday-Saturday	1h-7h

A.3.3.2 使用复合数据类型

可以通过组合基本和复合数据类型生成以下复合数据类型，从而生成新的数据类型：

- 数组(数据类型ARRAY)：一个数组是组合一组同一类型的数据形成一个单元。
- 结构体(数据类型STRUCT)：一个结构体组合不同的数据类型形成一个单元。
- 字符串(数据类型STRING)：一个字符串是义了一个最多有254字符(数据类型CHAR)的一维数组。字符串的长度必须与块的形参和实参相匹配。
- 日期和时间(数据类型DATE_AND_TIME)：日期和时间数据类型存储年、月、日、小时、分、秒、毫秒和星期。

下图所示说明数组和结构体是如何在一个区域内构造数据类型并存储信息的。可以在一个 DB 或一个 FB、OB 或 FC 的变量声明中定义一个数组或一个结构体。



A.3.3.3 使用数组访问数据

数组

一个数组组合一组同一类型的数据(基本类型或复合类型)形成一个单元。可以生成一个包含数组的数组，当定义一个数组时，必须作以下各项：

- 给数组指定一个名字。
- 用关键字ARRAY声明一个数组。
- 用下标指定数组的大小，在数组中指定每一维(最多6维)的第一个和最后一个数字。在一个方括号中输入下标，每一维之间用逗号隔开，该维的第一个数和最后一个数之间是两个点号。例如，下列下标定义了一个三维数组：

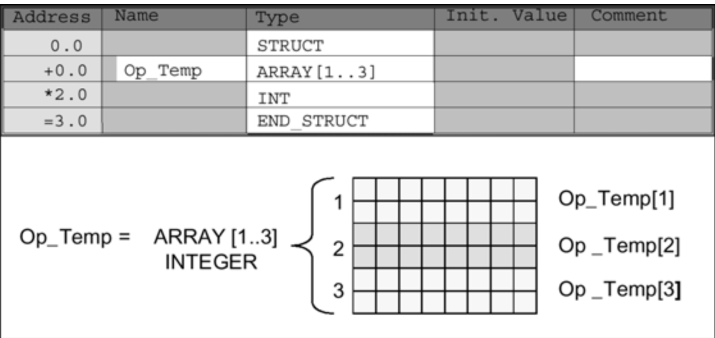
[1..5，-2..3，30..32]

- 指定数组中包含的数据的数据类型。

示例：1

下图所示是一个有三个整数的数组。使用下标可以访问存储在数组中的数据。下标就是方括号中的数字。例如，第二个整数的下标是 Op_temp[2]。

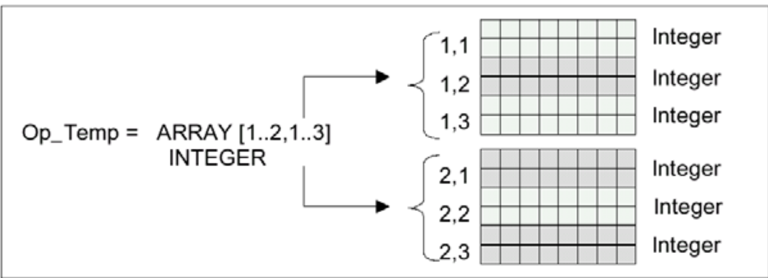
下标可以是包括负值在内的任意整数(-32768 到+32767)。下图中的数组也可以定义为 ARRAY[-1..1]。第一个整数的下标则为 Op_temp[-1]，第二个是 Op_temp[0]第三个整数则是 Op_temp[1]。



Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Op_Temp	ARRAY[1..3]		
*2.0		INT		
=3.0		END_STRUCT		

示例 2

一个数组还可以描述多维组的数据类型。下图所示为整数(integer)的二维数组。



可以使用下标访问多维数组的数据。在本例中，第一个整数是 Op_Temp[1，1]，第三个是 Op_Temp[1，3]，第四个是 Op_Temp[2，1]，第六个是 Op_Temp[2，3]。

一个数组最多可以定义到六维。例如，可以按如下所示将变量 Op_Temp 定义为六维数组：

ARRAY[1..3，1..2，1..3，1..4，1..3，1..4]

这个数组的第一个元素的下标是 Op_Temp[1，1，1，1，1，1]。最后一个元素的下标为 Op_Temp[3，2，3，4，3，4]。

生成数组

在 DB 块中或变量声明表中可以定义数组。在创建数组时要指定关键字(ARRAY)及其后方括号中的大小，如下所示：

[低限值..高限值]

在多维数组中还要指定其它的高限和低限值并用逗号隔开各维。下图所示为生成一个 2×3 数组的声明。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Heat_2x3	ARRAY[1..2,1..3]		
*2.0		INT		
=6.0		END_STRUCT		

为数组输入初始值

当生成数组时可以为每个数组元素赋予一个初始值。STEP 7 提供两种输入初始值的方法：

- 单个值的输入：可以为每个数组元素指定一个对于数组数据类型是有效的数值。要按元素的顺序指定数值：[1，1]。记住每个元素必须用逗号彼此隔开。
- 指定重复系数：使顺序的元素具有相同的初值，可以为这些元素指定元素个数(重复系数)和初始值。输入重复系数的格式为X(y)，这里X是重复系数，y是要重复的数值。

如果使用上图中的数组声明，可以按如下所示为所有的六个元素指定初始值：17，23，-45，556，3342，0。也可以通过指定 6(10)将所有六个元素的初始值设为 10。还可以为前两个元素指定特定的值而将其余的四个元素指定为 0，如：17，23，4(0)。

访问数组中的数据

可以通过数组中特定元素的下标访问数组中的数据。下标和符号名一起使用。

例如：上图中声明的数组从 DB20(motor)的第一个字节开始，可以用如下地址访问数组的第二个元素：

Motor.Heat_2×3[1，2]。

使用数组作为参数

可以将数组作为参数进行传送。如果一个参数在变量声明中被声明为数组，则必须传送整个数组(不是单个元素)。而数组中的一个元素可以在调用块时赋值给一个参数，只要数组元素与参数的数据类型相应。

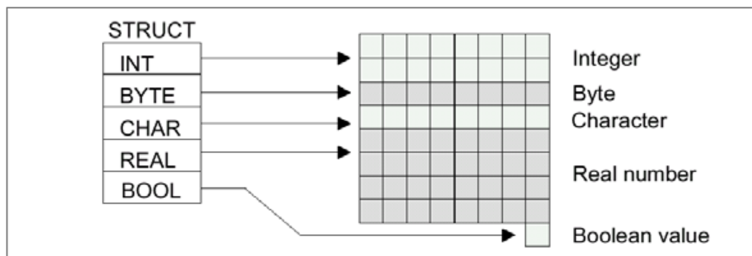
如果使用数组作为参数，数组无需与参数同名(它们甚至不需要名字)。而两个数组(形参和实参)的结构必须相同。例如，一个格式为 2×3 的整数数组，如果一个块的形参为 2×3 的整数数组，那么赋值的实参也必须为 2×3 的整数数组。

A.3.3.4　用结构体访问数据

结构体

结构体是组合各种数据类型(基本和复合数据类型，包括数组和结构体)形成一个单元。可以组合数据以适合过程控制。因此，可将参数作为一个数据单元来传送而不是单个元素。下图

所示为一个结构体，它包括一个整数、一个字节、一个字符、一个浮点数和一个布尔值。



一个结构体最多可有八层嵌套(例如，一个结构体包括包含数组的结构体)。

生成结构体

当在 DB 中或在一个逻辑块的变量声明中定义结构体。

下图是一个结构体(Stack_1)的声明，该结构体包括以下元素：一个整数(用于存储总量)，一个字节(用于存储原始数据)，一个字符(用于存储控制码)，一个浮点数(用于存储温度)以及一个布尔存储位(用于结束信号)。

Address	Name	Type	Init,Value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.1	End	BOOL	FALSE	
=10.0		END_STRUCT		

为一个结构体分配初始值

如果要给一个结构体的每个元素分配初始值，需要指定一个该数据类型的有效值以及元素的名字。例如，分配以下的初始值(给上图中声明的结构体)：

Amount	=	100
Original_data	=	B#(0)
Control_Code	=	`C'
Temperature	=	120
End	=	False

在结构体中存储和访问数据

可以访问结构体中的单个元素。可以使用符号地址(如，Stack_1.Temperature)。也可以指定元素所在的绝对地址(例如，如果 Stack_1 位于 DB20 的起始字节 0，amount 的绝对地址是 DB20.DBW0，temperature 的绝对地址是 DB20.DBD6)。

用结构体作为参数

可以传送作为参数的结构体。如果在变量声明中一个参数被声明为 STRUCT(结构体)，必

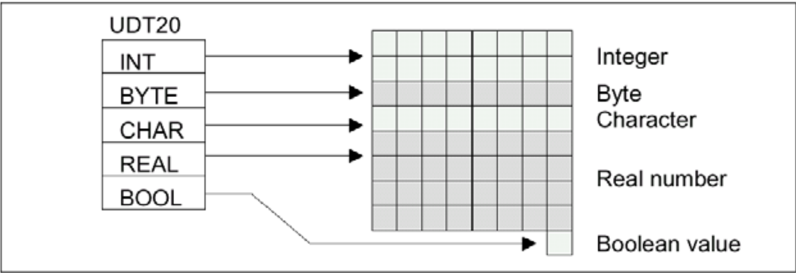
须为它传送一个具有相同元素的结构体。而一个结构体中的元素也可以被赋值给所用块的参数，只要结构体中的元素与参数的数据类型相符。

如果使用结构体作为参数，两个结构体(形参和实参)必须具有相同的元素，也就是说相同的数据类型按相同的顺序排列。

A.3.3.5 使用用户定义的数据类型访问数据

用户定义的数据类型

用户定义的数据类型(UDT)可以包括基本的和复合的数据类型。可以为 UDT 指定一个名字并且可以不止一次地使用它们。下图说明一个用户定义的数据类型的结构，它包括一个整数、一个字节、一个字符、一个浮点数和一个布尔值。



不用单个地或作为一个结构体输入所有的数据类型，只需要指定“UDT20”作为一个数据类型，STEP 7 自动地分配相应的存储空间。

生成一个用户定义的数据类型

用 STEP 7 可以定义 UDT。下图所示是一个 UDT，它包含以下元素：

一个整数(用于存储总量)、一个字节(用于存储原始数据)、一个字符(用于存储控制码)、一个浮点数(用于存储温度)以及一个布尔存储位(用于结束信号)。你可以在符号表中给 UDT 赋予一个符号名(如，process data)。

Address	Name	Type	Init,Value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.1	End	BOOL	FALSE	
=10.0		END_STRUCT		

一旦生成了一个 UDT，可以用 UDT 作为一个数据类型，例如，在一个 DB(或在一个 FB 的变量声明)中一个变量声明数据类型 UDT200。

下图所示是一个含有变量 process_data_1 的 DB，该变量的数据类型是 UDT200。只需要指定 UDT200 和 process_data_1。当编译该 DB 时，显示为斜体的数组被生成。

Address	Name	Type	Init,Value	Comment
0.0		STRUCT		
+~~6~~10.0	Process_data_1	UDT200		
=10~~6~~.0		END_STRUCT		

为用户定义的数据类型分配初值

如果要给用户定义的数据类型的每个元素分配初值，需要指定一个该数据类型的有效值以及元素的名字。例如，可以分配以下初值(给上图中声明的用户定义的数据类型)：

Amount	=	100
Original_data	=	B#16#0)
Control_code	=	\`C'
Temperature	=	1.2000000+002
End	=	False

如果声明了一个变量为 UDT，这个变量的初值就是生成 UDT 时指定的数值。

存储和访问用户定义的数据类型

可以访问一个 UDT 的每一个元素。可以使用符号地址(如，Stack_1.　　Temperature) 。也可以指定元素所在的绝对地址(例如，如果 Stack_1 位于 DB20 的起始字节 0，则 amount 的绝对地址是 DB20.DBWO 而 temperature 是 DB20.DBD6)。

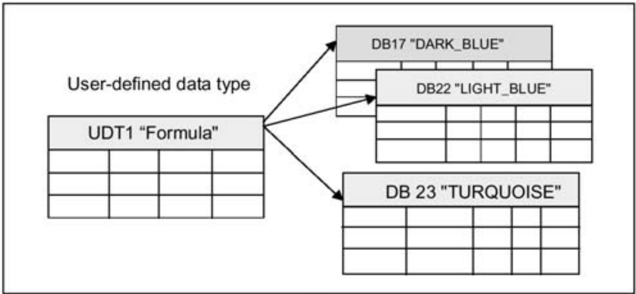
使用用户定义的数据类型作为参数

可以传送作为参数的 UDT 变量。如果一个参数在变量声明中声明为 UDT，必传送一个具有相同结构的 UDT。而当调用一个块的时候，UDT 的一个元素也可以赋值给一个参数，只要 UDT 的元素与参数的数据类型相符。

使用选定的 UDT 的 DB 的优点

使用一个一次生成的 UDT，可以生成许多具有相同数据结构的数据块。然后可以使用这些数据块为特定的任务存入不同的实际值。

例如，为一个配方(如调和颜色)构造一个 UDT，可以将这个 UDT 指定给几个 DB，每个包含不同数量。



数据块的结构由指定给定的 UDT 决定。

A.3.4 参数类型

除了基本和复合数据类型之外，还可以为块之间传递的形参定义参数类型。STEP 7 可以识别以下参数类型：

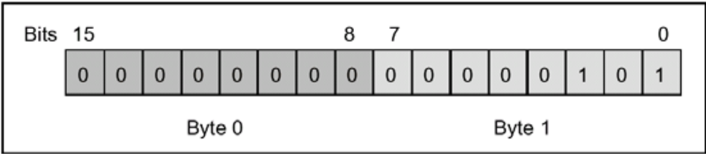
- TIMER(定时器)或COUNTER(计数器)：这里指定块执行时将使用一个特定的定时器或计数器。如果将TIMER或COUNTER参数类型作为形参，则相应的实参也必须是一个定时器或一个计数器，换句话说，输入一个后面跟着正整数的“T”或“C”。
- BLOCK(块)：指定一个特定的块用作输入或输出。由参数的声明决定使用的块的类型(FB、FC、DB等)。如果为BLOCK参数类型的形参提供数值，要指定一个块地址作为实参。例如“FC101”(当使用绝对地址时)或“Value”(使用符号地址)。
- POINTER(指针)：指向一个变量的地址。指针包含一个地址而不是一个数值。当为一个PONITER参数类型的形参提供数值时，必须指定一个地址作为实际参数。在STEP 7中，可以用指针的格式来指定一个指针，也可以简单地用一个地址来指定(如，M50.0)。以指针的格式寻址从M50.0开始的数据可以写为：P#M50.0
- ANY：这一类型用于实参的数据类型不知道或可以是任何数据类型的情形。要了解关于ANY参数类型的更多的信息，可参考“参数类型ANY的格式”以及“使用参数类型ANY”。

参数类型也可用于用户定义的数据类型(UDT)。有关 UDT 的更多的信息参考”使用用户定义的数据类型访问数据。”

参数	容量	描述
TIMER	2字节	指示在被调用的逻辑块中由程序使用的定时器 格式：　T1
COUNTER	2字节	指示在被调用的逻辑块中由程序使用的计数器 格式：　C10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2字节	指示在被调用的逻辑块中由程序使用的块 格式：　FC101 DB42
POINTER	6字节	标识地址 Format:　P#M50.0
ANY	10字节	用于当前参数的数据类型是未知的 格式：　P#M50.0 BYTE 10　数据类型是ANY格式 P#M100.0 WORD 5 L#1COUNTER 10　参数类型是ANY格式

A.3.4.1 参数类型 BLOCK、COUNTER、TIMER 的格式

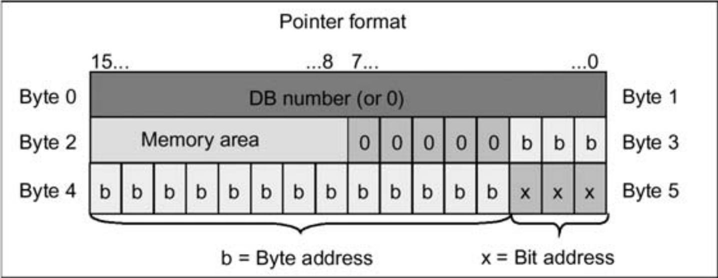
STEP 7 将参数类型 BLOCK、COUNTER 和 TIMER 作为二进制数存在一个字中(32 位)。下图所示为这些参数类型的格式。



允许使用的块、定时器和计数器的数量根据 S7 CPU 的类型而定。在“S7-300 可编程控制器、硬件和安装手册”或“S7-400，M7-400 可编程控制器，硬件和安装手册”的数据页中能够找到更多关于 CPU 能够使用的定时器、计数器的数目以及可用块的最大数目的信息。

A.3.4.2 参数类型 POINTER 的格式

下图所示为该类型的数据在每一个字节中的存储。



参数类型 POINTER 存储以下信息：

- DB号码(如果数据未存储在DB块中则为0)
- CPU中的存储区域(下表所示为参数类型POINTER的存储区域的十六进制代码)

十六进制代码	存储区域	描述
B#16#81	I	输入区域
B#16#82	Q	输出区域
B#16#83	M	位存储区域
B#16#84	DB	数据块
B#16#85	DI	背景数据块
B#16#86	L	局域数据(L堆栈)
B#16#87	V	上一个块调用使用的局域数据

- 数据的地址(字节.位的格式)

STEP 7 提供指针格式：P#memory_area byte.bit_address(P#存储区域 字节.位地址)(如果形参被声明为参数类型 POINTER，只需指出存储区域和地址。STEP 7 会自动将输入转换为指针的格式)。下面的例子说明如何为从 M50.0 开始的数据输入参数类型 POINTER：

- P#M50.0
- M50.0(如果形参声明的POINTER)。

A.3.4.3 使用参数类型 POINTER

指针用于指向一个地址。这种寻址方式的优势在于可以在程序执行过程中动态地修改语句的地址。

指针用于存储器间接寻址

用存储器间接寻址的程序语句由一个指令，一个地址标识和一个偏移量组成。(该偏移量必须在方括号中给出)。

双字格式的指针举例：

L	P#8.7	将指针数值装入累加器1
T	MD2	将指针传入MD2
A	I[MD2]	查询输入位I8.7的信号状态并且
=	Q[MD2]	将信号状态赋给输出位Q8.7

指针用于区域内部和区域交叉寻址

用这种寻址类型的程序指令由一个指令和以下部分组成：地址标识、地址寄存器标识符、偏移量。地址寄存器(AR1/2)和偏移量必须在方括号内一起被指定。

区域内部寻址示例

指针中不包含存储区域的指示：

L	P#8.7	将指针数值装入累加器1
LAR1		将指针从累加器1装入AR1
A	I[AR1,P#0.0]	查询输入位I8.7的信号状态并且
=	Q[AR1,P#1.1]	将信号状态赋给输出位Q10.0

偏移量 0.0 不产生任何影响，输出 10.0 由 8.7(ARI)加偏移量 1.1 计算得出。结果为 10.0 而不是 9.8，见指针格式。

区域交叉寻址示例

在区域交叉寻址中指针指示存储区域(在示例中为 I 和 Q)。

L	P#I8.7	装载指针值和区域标识到累加器1
LAR1		装载存储区域I和地址8.7到AR1
L	P#Q8.7	装载指针值和区域标识到累加器1
LAR2		装载存储区域Q和地址8.7到AR2
A	[AR1,P#0.0]	查询输入位I8.7的信号状态并且
=	[AR1,P#1.1]	将信号状态赋给输出位Q8.7

偏移量 0.0 不产生影响。输出 10.0 由 8.7(AR2)加偏移量 1.1 计算得出。结果是 10.0 而不是 9.8，见指针格式。

A.3.4.4 用于修改指针的块

使用示例块 FC3“Routing Pointers”可以修改一个指针的位地址或字节地址。当该 FC 被调用时需要修改的指针传送给变量“Pointer”(可以使用双字格式的区域内部和区域交叉指针)。

用参数“Bit-Byte”可以修改指针的位或字节地址(0：位地址，1：字节地址)。变量“Inc-Value”(整数格式)指定要从地址内容中加上或减去的数值。也可以指定负值来减小地址。

改变位地址，会对字节地址产生进位(或在减少情况下)，例如：

- P#M5.3，Bit_Byte=0，Inc_Value=6 => P#M6.1或
- P#M5.3，Bit_Byte=0，Inc_Value=-6 => P#M4.5

指针的区域信息不受该功能影响。

FC 截取指针的上溢/下溢。这种情况下指针不变，输出变量“RET_VAL”(可以错误处理)设为“1”(直至下一个 FC3 的正确处理)。这里：

- 1.位地址被选中并且Inc_Value>7，或<-7
- 2.位或字节地址被选中并且修改会造成一个“负”字节地址
- 3.位或字节地址被选中，修改会造成一个非法的过大的字节地址。

修改指针的 STL 示例块

```
FUNCTION FC 3:BOOL
TITLE=Routing Pointers
//FC3 可用于修改指针
AUTHOR：AUTICS1
FAMILY:INDADDR
NAME:ADDRPOINT
VERSION:0.0

VAR_INPUT
```

```
    Bit_Byte:BOOL://0:位地址 1：字节地址
    Inc_Value:INT；//增量(如果数值为负=>减小/如果数值为正=>增加)
END_VAR

VAR_IN_OUT
    pointer:DWORD;//要修改的指针
END_VAR
VAR_TEMP
    inc_Value1:INT;//中间值增量
    pointer1:DWORD；//中间值指针
    Int_Value:DWORD;//辅助变量
END_VAR
BEGIN
NETWORK
TITLE=
//该块截取对指针区域信息的修改
//或自动转为负指针
    SET    ;//置 RLO 为 1 并且
    R      #RET_VAL;//复位溢出
    L      #Pointer;//为临时变量提供数值
    T      #Pointer1;//中间值指针
    L      #Inc_Value;//为临时变量提供数值
    T      #Inc_Value1;//中间值增量
    A      #Bit_Byte;//如果=1,字节地址指令
    JC     Byte;//跳转到字节地址的计算
    L      7;//如增量值>7
    L      #Inc_Value1;
    <I     ;
    S      #RET_VAL;//置位 RET_VAL 并且
    JC     End;//跳到 End
    L      -7;//如果增量值<-7
    <I     ;
    S      #RET_VAL;//置位 RET_VAL 并且
    JC     End;//跳转到 End
    A      L     1.3；//如果该值的位 4=1(Inc_Value 为负)
    JC     neg;//则跳转到位地址的减法计算
    L      #Pointer1;//L 装载指针地址信息
    L      #Inc_Value1;//加上增量
    +D     ；
    JU     test；//跳转到对负结果的测试
neg：      L     #Pointer1;//装载指针地址
    L      #Inc_Value1;//装载增量
    NEG1 ;//对负值取反
```

```
     -D    ；//减去增量值
     JU    test;//跳转到测试
Byte:      L      0;//开始字节地址修改
     L     #Inc_Value1;//如果增量>=0 则
     <I    ;
     JC    pos;//跳转到加法，否则
     L     #Pointer1;//装载指针地址信息
     L     #Inc_Value1;//装载增量
     NEGI；//对负值取反
     SLD   3;//将增量向左移 3 位
     -D    ；//减去该数值
     JU    test;//并跳转到测试
pos:       SLD  3;//增量左移三位
     L     #Pointer1;//装载指针地址信息
     +D    ;//加增量
test: T    #Int_Value;//将计算结果传至 Int_Value.
     A     L 7.3;//如果字节地址无效(太大或为负)，
     S     #RET_VAL;//则置位 RET_VAL
     JC    End;//并跳转到 End，
     L     #Int_Value;//否则传送结果
     T     #Pointer;//到指针
End:       NOP  0;
END_FUNCTION
```

A.3.4.5 参数类型 ANY 的格式

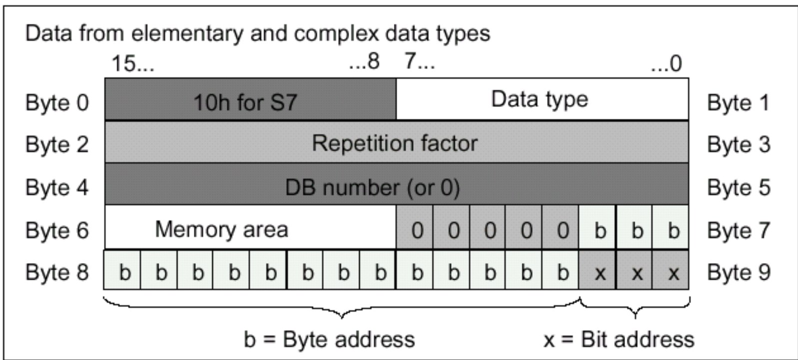
STEP 7 用 10 个字节(80 个位)来存储参数类型 ANY。当构造 ANY 参数类型时，由于被调用的块要评估该参数的整个内容，必须确保所有的 80 个位都被占用。例如，如果在字节 4 指定一个 DB 号码，还必须在字节 6 直接指定存储区域。

STEP 7 对基本数据和复合数据类型的管理与参数类型的数据不同。

ANY 格式的数据类型

对基本和复合数据类型，STEP 7 存储以下数据：

- 数据类型
- 重复系数
- DB号码
- 保存信息的存储区域
- 数据的起始地址



重复系数指明由参数类型 ANY 要传送的所指数据类型的数量。这意味着可以指定一个数据区域，并且用数组和结构体与参数类型 ANY 连接。STEP 7 将数组和结构体作为以字节为单位的数来识别(在重复系数的帮助下)。例如，如果要传送 10 个字，必须输入 20(字节)作为重复系数。

地址以字节.位的格式存储，其中字节地址存在字节 7 的 0 位至 2 位，字节 8 的 0 位至 7 位以及字节 9 的 3 位至 7 位。位地址存储在字节 9 的 0 位至 2 位。

对于类型为 NIL 的空指针，从字节 1 开始的所有字节都赋值为 0。

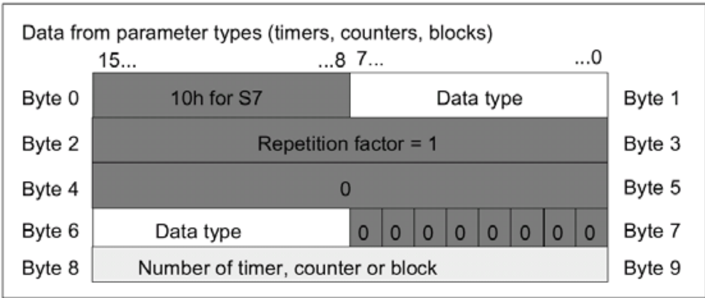
下表所示为参数类型 ANY 数据类型与 CPU 存储器的编码。

十六进制码	数据类型4码	描述
B#16#00	NIL	Null pointer空指针
B#16#01	BOOL	Bits位
B#16#02	BYTE	Bytes(8 bits)字节（8位）
B#16#03	CHAR	Characters(8 bits)字符（8位）
B#16#04	WORD	Words(16 bits)字（16位）
B#16#05	INT	Integers(16 bits)整数（16位）
B#16#06	DWORD	Words(32 bits)字（32位）
B#16#07	DINT	Double integers(32 bits)双整数（32位）
B#16#08	REAL	浮点数（32位）
B#16#09	DATE	Date日期
B#16#0A	TIME_OF_DAY(TOD)	Time of day日时钟
B#16#0B	TIME	Time时间
B#16#0C	S5TIME	Data type S5TIME数据类型S5TIME
B#16#0E	DATE_AND_TIME(DT)	Date and time(64 bits)日期和时间（64位）
B#16#13	STRING	String字符串

十六进制码	数据类型4码	描述
B#16#81	I	输入区
B#16#82	Q	输出区
B#16#83	M	位存储区
B#16#84	DB	数据块
B#16#85	DI	背景数据块
B#16#86	L	局部数据(L栈)
B#16#87	V	上一个块调用使用的局域数据

ANY 格式的参数类型

STEP 7 为参数类型存储数据类型和参数地址。重复系位总是 1。字节 4，5 和 7 总是 0，字节 8 和 9 指示定时器、计数器或块的号码。



下表所示为参数类型 ANY 的数据类型代码。

数据类型的代码		
十六进制代码	数据类型	描述
B#16#17	BLOCK_FB	FB号码
B#16#18	BLOCK_FC	FC号码
B#16#19	BLOCK_DB	DB号码
B#16#1A	BLOCK_SDB	SDB号码
B#16#1C	COUNTER	计数器号码
B#16#1D	TIMER	定时器号码

A.3.4.6 使用参数类型 ANY

可以为块定义一个形参使之适合任何数据类型的实参。当块被调用时其提供的实参的数据类型是未知的或可变的(以及允许是任意数据类型)时候，这种功能特别有用。在块的变量声明中，可以声明参数为数据类型 ANY。然后在 STEP 7 中可以指定任意数据类型的一个实参。

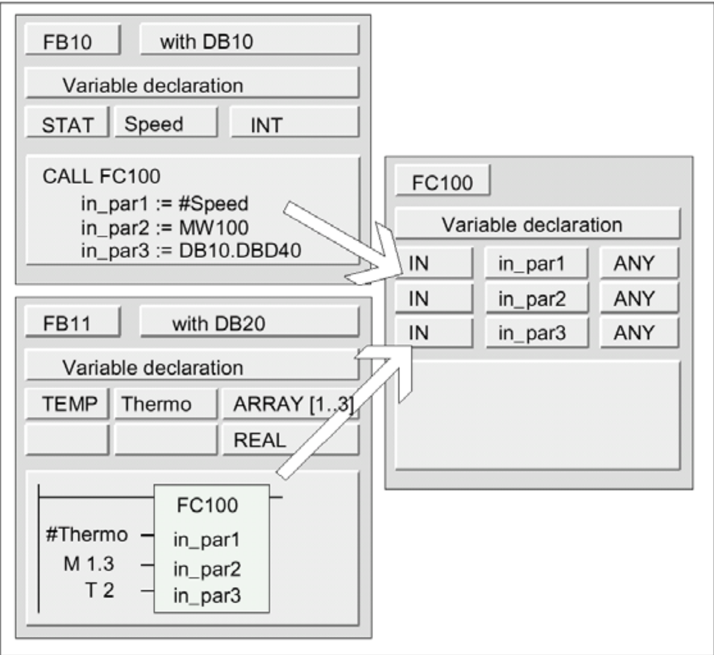
STEP 7 为 ANY 数据类型的变量分配 80 个位的存储区。如果将一个实参赋值给这个形参，STEP 7 会将实参的起始地址、数据类型和长度编码在这 80 个位中。被调用的块分析为这个 ANY 参数所存储的 80 个位的数据，获得进一步处理所需的信息。

分配一个实参给一个 ANY 参数

如果一个参数数据类型为 ANY,则可以分配一个任意数据类型的实参给这个形参。在 STEP 7 中，可以指定以下数据类型作为实参：

- 基本数据类型：可以指定这个实参的绝对地址或符号名。
- 复合数据类型：对于复合数据类型(如数组和结构体)的数据要指定符号名。
- 定时器、计数器和块：指定号码(例如，T1，C20或FB6)。

下图说明数据是如何传送给一个带有数据类型为 ANY 参数的 FC。



在本例中 FC100 有三个参数(in_parl,in_par2 以及 in_par3)被声明为 ANY 数据类型。

- 当FB10调用FC100时，FB10传送一个整数(静态变量speed)，一个字(MW100)以及一个双字到DB10(DB10.DBD40)。
- 当FB11调用FC100时，FB11传送一个实数数组(临时变量“Thermo”)，一个布尔值(M1.3)，以及一个定时器(T2)。

为一个 ANY 参数指定一个数据区域

不仅可以给一个 ANY 参数指定一个单个地址(例如，MW100)，而且还可以指定一个数据区域。如果想指定一个区域作为实参，使用下列常数格式指定要传送的数据的量：

P#　　区域标识 字节.位　　数据类型　重复系数

对于数据类型，可以以常数的格式指定所有的基本数据类型和数据类型 DATE_AND_TIME。如果数据类型不是 BOOL，位地址必须指定 0(x.0)。下表是对三个格式示例的说明，这些格式用于指定传送给一个 ANY 参数的存储区域。

实参	描述
P#M50.0 BYTE 10	在字节存储区域指定10个字节： MB50至MB59
P#DB10.DBX5.0 S5TIME 3	指定三个位于DB10中的数据类型为 S5TIME的数据单位：DB字节5至DB字节10
P#Q10.0 BOOL 4	在输出区域指定4位：Q10.0至Q10.3

使用参数类型 ANY 的示例

下例说明如何使用参数类型 ANY 和系统功能 SFC20 BLKMOV 复制一个 10 个字节的存储区域。

STL	解释
FUNCTION FC10:VOID	
VAR_TEMP	
Source : ANY;	
Target : ANY;	
END_VAR	
BEGIN	
LAR1 P#Source;	将 ANY 指针的起始地址装入 AR1。
L B#16#10;	装载语法标识(ID)并且
T LB[ARI,P#0.0];	传送给 ANY 指针
L B#16#02；	装载数据类型字节并且
T LB[AR1,P#1.0]；	传送给 ANY 指针
L 10;	装载 10 个字节并且
T LW[AR1,P#2.0]；	传送给 ANY 指针
L 22；	源是 DB22，DBB11
T LW[AR1,P#4.0]；	
L P#DBX11.0；	
T LD[AR1,P#6.0]；	
LAR1 P#Target;	装载 ANY 指针的起始地址到 AR1
L B#16#10;	装载语法标识(ID)并且
T LB[AR1,P#0.0]；	传送给 ANY 指针
L B#16#02；	装载数据类型字节并且
T LB[AR1,P#1.0]；	传送给 ANY 指针
L 10;	装载 10 个字节并且
T LW[AR1,P#2.0]；	传送给 ANY 指针
L 33;	目标是 DB33，DBB202
T LW[AR1,P#4.0]；	
L P#DBX202.0；	
T LD[AR1,P#6.0];	
CALL SFC 20(	调用系统功能 BLKMOV
SRC BLK:=Source,	
RET_VAL:=MW12，	评估 BR 位和 MW12
DSTBLK:=Target	
)；	
END_FUNCTION	

A.3.4.7 为逻辑块的局域数据指定数据类型

使用 STEP 7，在块变量声明中，分配给局域数据的数据类型(基本和复合数据类型以及参数类型) 有限定的。

OB 局域数据的有效数据类型

下表说明为一个 OB 声明局域数据的限定(一)。由于不可能调用一个 OB，所以 OB 没有参数(输入、输出或入/出)。由于 OB 没有背景 DB，所以不能为 OB 声明任何静态变量。OB 的临时变量的数据类型可以是基本的或复合的数据类型以及数据类型 ANY。

有效的赋值用符号 • 表示。

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
Input	—	—	—	—	—	—	—
Output	—	—	—	—	—	—	—
In/out	—	—	—	—	—	—	—
Static	—	—	—	—	—	—	—
Temporary	•(1)	•(1)	—	—	—	—	•(1)

1）位于OB的L堆栈中。

FB 局域数据的有效数据类型

下表说明为一个 FB 声明局域数据的限定(一)。由于使用背景 DB，为 FB 声明局域数据所受的限定很少。当声明输入参数时没有任何限定；对于一个输出参数不能声明任何参数类型，而对于入/出参数所允许的参数类型只有 POINTER 和 ANY。可以声明临时变量为 ANY 数据类型，其它的所有参数类型都是非法的。

有效的赋值用符号 • 表示。

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTE R	BLOCK	POINTE R	ANY
Input	•	•	•	•	•	•	•
Output	•	•	—	—	—	—	—
In/out	•	•(1)，(3)	—	—	—	•	•
Static	•	•	—	—	—	—	—
Temporary	•(2)	•(2)	—	—	—	—	•(2)
（1）在背景数据块中存为参考(48位指针)							
（2）位于FB的L推栈中。							
（3）字符串只能定义为却省长度。							

FC 局域数据的有效数据类型

下表说明为一个 FC 声明局域数据的限定(一)。由于 FC 没有背景 DB，所以它没有静态变量。对于 FC 的输入、输出和入 / 出参数只有参数类型 POINTER 和 ANY 是允许的。也可以声明临时变量为 ANY 参数类型。

有效的赋值用符号 • 表示。

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
Input	•	•(2)	•	•	•	•	•
Output	•	•(2)	—	—	—	•	•
In/out	•	•(2)	—	—	—	•	•
Static	—	—	—	—	—	—	—
Temporary	•(1)	•(1)	—	—	—	—	•(1)

（1）位于FC的L的堆栈

（2）字符串只能定义为却省长度。

A.3.4.8 程序块间传送参数时允许的数据类型

程序块之间参数传送的规则

当给形参赋实参时，可以指定一个绝对地址\一个符号名或一个常数。STEP 7 为不同的参数限定有效的赋值。例如，输出和输入/输出参数不能被指定为常数(因为输出或入/出参数的目的就是要改变它的数值)。这些限定特别地应用于复合数据类型的参数，对它们既不能指定绝对地址也不能是常数。

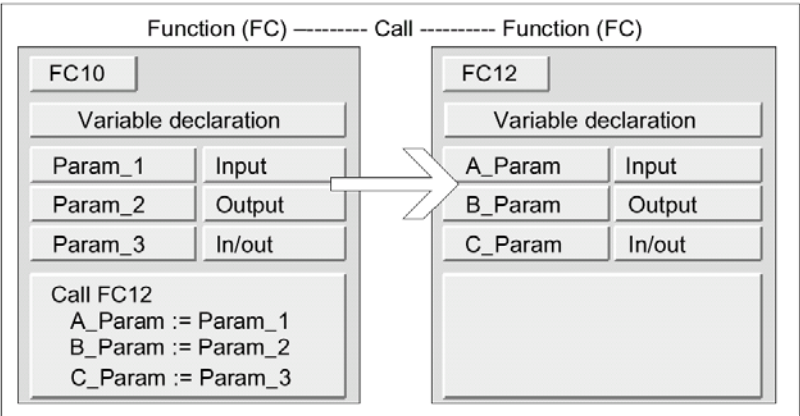
下表说明对涉及给形参赋实参的数据类型的限定。

有效的赋值使用符号 • 表示。

基本数据类型				
声明类型	绝对地址	符号名（在符号表中）	临时局域符号	常数
Input	•	•	•	•
Output	•	•	•	—
In/out	•	•	•	—
复合数据类型				
声明类型	绝对地址	DB元素的符号名	临时局域符号	常数
Input	—	•	•	—
Output	—	•	•	—
In/out	—	•	•	—

功能（FC）调功能（FC）时的有效数据类型

可以将一个调用 FC 的形参赋值给一个被调用的 FC 的形参。下图说明将 FC10 的形参作为实参赋值给 FC12 的形参。



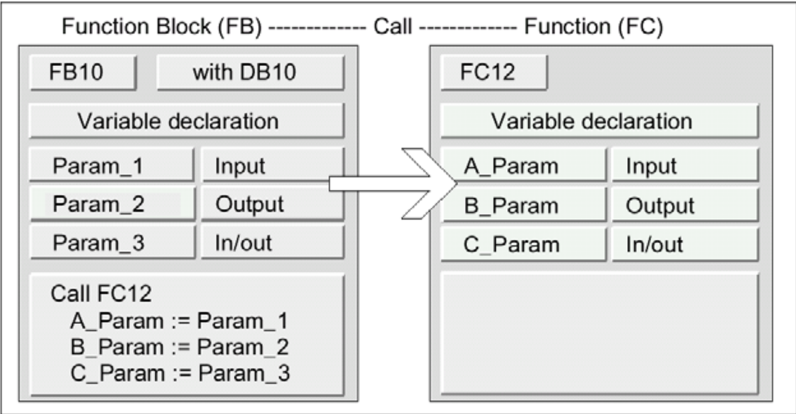
STEP 7 对于将一个 FC 的形参作为实参赋值给另一个 FC 的形参作了限定。例如，不能将复合数据类的参数或一个参数类型作为实参赋值。

下表所示为一个 FC 调用另一个 FC 时允许的数据类型(•)。

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
Input→Input	•	—	—	—	—	—	—
Input→Output	—	—	—	—	—	—	—
In/out→In/out	—	—	—	—	—	—	—
Output→Input	—	—	—	—	—	—	—
Output→Output	•	—	—	—	—	—	—
Output→In/out	—	—	—	—	—	—	—
In/out→Input	•	—	—	—	—	—	—
In/out→Output	•	—	—	—	—	—	—
In/out→In/out	•	—	—	—	—	—	—

功能块（FB）调用功能（FC）时的有效数据类型

可以将一个调用 FB 的形参赋值给一个被调用的 FC 的形参。下图说明将 FB10 的形参作为实参赋值给 FC12 的形参。

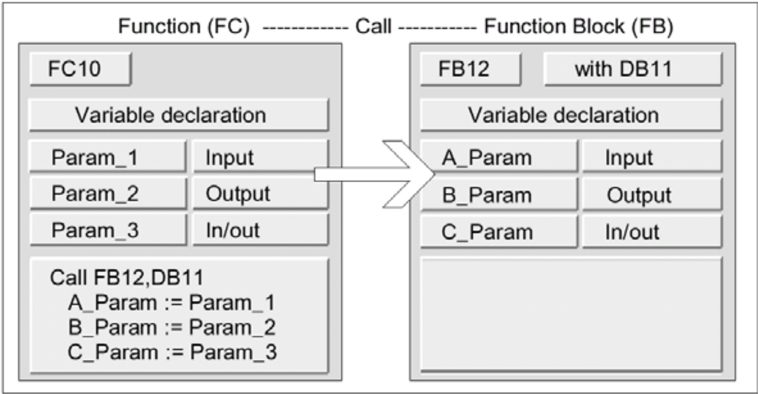


STEP 7 对于将一个 FB 的形参赋值给一个 FC 的形参作了限定。例如，不能将参数类型的参数作为实参赋值。下表所示为一个 FB 调用一个 FC 时允许的数据类型(•)。

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
Input→Input	•	•	—	—	—	—	—
Input→Output	—	—	—	—	—	—	—
Input→In/out	—	—	—	—	—	—	—
Output→Input	—	—	—	—	—	—	—
Output→Output	•	•	—	—	—	—	—
Output→In/out	—	•	—	—	—	—	—
In/out→Input	•	—	—	—	—	—	—
In/out→Output	•	—	—	—	—	—	—
In/out→In/out	•	—	—	—	—	—	—

功能（FC）调用功能块时的有效数据类型

可以将一个调用 FC 的形参赋值给一个被调用的 FB 的形参。下图说明 FC10 的形参作为实参赋值给 FB12 的形参。



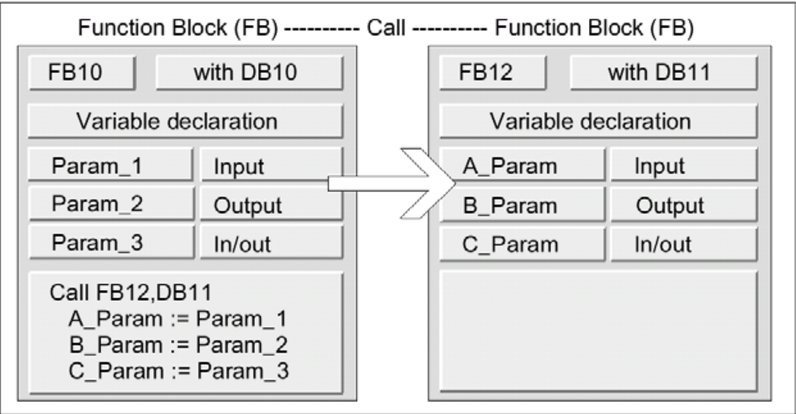
STEP 7 对将一个 FC 的形参赋值给一个 FB 的形参作了限定。例如，不能将复合数据类型的参数作为实参赋值。但是，可以将参数类型为 TIMER、COUNTER 或 BLCCK 的输入参数赋值给被调用 FB 的输入参数。

下表所示为一个 FC 调用一个 FB 时允许的数据类型(•)。

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
Input→Input	•	—	•	•	•	—	—
Input→Output	—	—	—	—	—	—	—
Input→In/out	—	—	—	—	—	—	—
Output→Input	—	—	—	—	—	—	—
Output→Output	•	—	—	—	—	—	—
Output→In/out	—	—	—	—	—	—	—
In/out→Input	•	—	—	—	—	—	—
In/out→Output	•	—	—	—	—	—	—
In/out→In/out	•	—	—	—	—	—	—

功能块调用功能块时的有效数据类型

可以将一个调用 FB 的形参赋值给一个被调用的 FB。下图说明 FB10 的形参作为实参赋值给 FB12 的形参。



STEP 7 对于将一个 FB 的形参赋值给另一个 FB 的形参作了限定。例如，不能用复合数据类型的输入和输出参数作为实参赋值给被调用的 FB 的输入和参数。但是可以将参数类型为 TIMER、COUNTER 或 BLOCK 的输入参数赋值给被调用的 FB 的输入参数。

下表所示为一个 FB 调用另一个 FB 时允许的数据类型(•)

声明类型	基本数据类型	复合数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
Input→Input	•	•	•	•	•	—	—
Input→Output	—	—	—	—	—	—	—
Input→In/out	—	—	—	—	—	—	—
Output→Input	—	—	—	—	—	—	—
Output→Output	•	•	—	—	—	—	—
Output→In/out	—	—	—	—	—	—	—
In/out→Input	•	—	—	—	—	—	—
In/out→Output	•	—	—	—	—	—	—
In/out→In/out	•	—	—	—	—	—	—

A.3.4.9 向功能块的 IN_OUT 参数作传送

当复合数据类型被传送给功能块(FB)的 IN_OUT 参数时，变量的地址被传送(由参考调用)。

当基本数据类型被传送给一个功能块的 IN_OUT 参数时，在起动功能块之前数值被复制到背景数据块中，当功能块结束后又从背景数据复制回来。这意味着基本数据类型的 IN_OUT 变量可以被一个数值初始化，但是，在调用中不能用一个常数作为实参代替 IN_OUT 变量，因为不能对一个常数作写操作。

数据类型为 STRUCT 或 ARRAY 的变量不能被初始化，因为这种情况在背景数据块中只有一个地址。

A.4 使用旧项目工作

A.4.1 转换版本 1 的项目

对于使用 STEP 7 版本 1 生成的项目可以再利用。如需这样，必须将版本 1 的项目转换为版本 2 的项目。

版本 1 的项目中以下组件可以保留：

- 带有程序的项目结构
- 块
- STL源文件
- 符号表

硬件组态不会被转换。不能复制这个项目中的程序组件到其它的项目。还可以在新项目中增加一个站并对它组态和参数赋值。

一旦完成版本 2 的转换，就可以在一个对话框中决定现在是否要把这个版本 2 的项目转换为当前版本下 STEP 7 的项目。

提示

各个块仍保留版本 1 块作为它们的属性。版本 1 中生成的程序不会修改，因此不能被用来连接多重背景。

如果想在转换过来的块中声明多重背景，首先使用“LAD/STL/FBD：Programming BLocks”应用程序将转换过来的块生成 STL 源文件，然后再编译源文件，生成新的块替代原有的块。

编程多重背景是 STEP 7 版本 2 的新特性，用以生成功能块(FB)。如果想在版本 2 的项目中按原来的方式继续使用在版本 1 中生成的功能块，不必转换它们。

步骤

要转换版本 1 的项目，按如下进行

1. 选择菜单命令 File > Open Version 1 Project
2. 在出现的对话框中选择要转换为版本 2 的项目，可以通过扩展名*.s7a 来识别版本 1 的项目。
3. 在下一个对话框中，为版本 1 的项目转换后的新项目输入名字。

A.4.2 转换版本 2 项目

在 STEP 7 中也可以用菜单命令 **File > Open** 打开版本 2 的项目。

版本 2 的项目/库可以被转换(迁移)到你当前的 STEP 7 版本。使用菜单命令 **File > Save As** 并选中选项”Rearrage before saving(存储前重新整理)”，该项则被存为当前 STEP 7 版本下的一个项目。

可以编辑来自旧的 STEP 7 版本的项目和库并保持它们的格式，并且在“Save Project As”对话框中选择旧的 STEP 7 版本作为它们保存的文件类型。例如，要编辑 STEP 7 版本 2.1 的对象，这里选择“Project 2.x”或“library 2.x”（不可能将 5.1 以上的版本存储为版本 2，可以参考编辑版本 2 的项目和库）。

文件类型标志

	STEP 7 V3	从STEP 7 V4开始
当前版本的文件类型	Project 3.x Library 3.x	Project Library
旧版本的文件类型	Project 2.x Library 2.x	Project 2.x Library 2.x

这意味着只能访问某一范围内的 STEP 7 旧版本的功能。但是仍可以继续用旧的 STEP 7 版本管理这些项目和库。

提示

从版本 3 更新到版本 4 及更高的版本只涉及名字的变化：格式仍保持一致。因此在 STEP 7 V4 中不存在文件类型“Project3.x”。

步骤

将版本 2 的项目转换为当前 STEP 7 版本的格式可按如下进行：

1. 在 File 菜单中为该项目执行“Save As”命令以及“Rearrange before saving”选项。
2. 在“Save Project As”对话框中选择文件类型“Project”并点击“Save”按钮。

要转换版本 2 的项目到当前的 STEP 7 版本同时要保持它们的格式，可按如下进行：

1. 如果有必要执行以上步骤 1。
2. 在“Save Praject As”对话框中选择旧的 STEP 7 版本的文件类型并点击“Save”按钮。

A.4.3 关于 STEP 7 V2.1 项目使用 GD 通讯的提示

- 如果要将一个使用全局数据的STEP 7 V2.1的项目转换为STEP 7 V5，首先在STEP 7 V2.1的项目中用STEP 7V5.0打开GD表。以前所组态的通讯数据通过GD通讯应用程序自动转换为新的结构。
- 当存档一个STEP 7 V2.1项目时，如果项目中包含文件名长于八个字符的文件时，旧程序(ARJ，PKZIP…)会发出一个错误信息。如果被编辑的STEP 7 V2.1项目中的MPI网络具有长于8个字符的ID(标识)时，也会出现这条错误信息。在首次启动组态全局数据通信之前，在有全局数据的STEP 7 V2.1项目中，为MPI网络编辑一个不超过8个字符长的名字。
- 如果要更改一个STEP 7 V2.1项目名，必须在GD表的标头栏(CPU)中通过重新选择适当的CPU来重新赋值。如果恢复旧的项目名，该赋值关系会被再次显示。

A.4.4 DP 从站丢失 GSD 文件或 GSD 文件有故障

如果用 STEP 7 V5.1 处理以前组态的站，可能会丢失 DP 从站的 GSD 文件包，或不能编译 GSD 文件（例如由于 GSD 文件中有语法错误）

在这种情况下，STEP 7 生成一个“虚假”的从站来代表所组态的从站，例如当一个站复制到编程器中或打开项目执行更多的处理。该“虚假”从站只能做有限的处理，不能更改从站结构（DP 标识符）以及从站参数。从站原有组态被保留，也可以删除整个从站。

重新对 DP 从站组态并赋值参数

如果要对 DP 从站进行重新组态和参数赋值，必须从制造商申请最新的 GSD 文件并通过 **Options>Install New GSD** 进行安装。

装入正确文件后，用它来代表 DP 从站。此时 DP 从站包含了所需数据，并可以对其进行处理。

A.5 示例程序

A.5.1 示例项目和示例程序

安装 CD 中包含许多示例项目。在 SIMATIC Manager 的“Open”对话框中可以找到这些示例项目。当安装选项软件包时可能还会有其它示例项目。有关这些示例项目的信息请参见选项软件包中的资料。

实例和示例项目	包括在CD中	在本章中已描述	OB1中的说明
“Zen01_01_STEP7_*” to “Zen01_06_STEP7_*”项目（使用入门和练习）	•	单独的手册	•
“Zen01_11_STEP7_DezP”项目（PROFIBUS DP组态实例）	•	-	-
“Zen01_08_STEP7_Blending”项目（工业混料实例）	•	•	-
“Zen01_09_STEP7-Zebra”项目（斑马线交通信号控制）	•		•
“Zen01_10_STEP7_COM-SFB”项目（两个S7-400 CPU间的数据交换）	•		•
“ZXX01_14_Hsystem_S7400H”项目（容错系统的起动项目）	•	单独的手册	•
“ZXX01_15_Hsystem_RED_IO”项目（带冗余I/O设备的容错系统的起动项目）	•	单独的手册	•
“Zen01_11_STEP7_COM_SFC1” 和 ”Zen01_11_STEP7_COM_SFC12”项目（为非组态的连接用通讯SFC交换数据）	•		•
项目 “ZEn01_13_STEP7_PID-Temp” (使用FB 58 和FB59进行温度控制的示例)	•		•
处理日时钟中断实例		•	
处理时间延迟中断实例		•	
屏蔽和中断屏蔽同步错误的实例		•	
禁止和使能中断及异步错误的实例		•	
中断和异步错误的延迟处理实例		•	

这些实例的侧重点不是要教会一种控制特定过程所需的编程样式或专家知识，这些实例只不过用来说明设计一个程序应遵循的步骤。

删除和安装提供的示例项目

提供的示例项目在SIMATIC管理器中可以被删除，然后还可以再安装。要安装示例项目，必须起动STEP 7 V5.0的安装程序，示例项目也可以被选则稍后安装。可以将提供的示例程序及用户生成的示例程序使用菜单命令“Save As”存储为用户项目。

提示

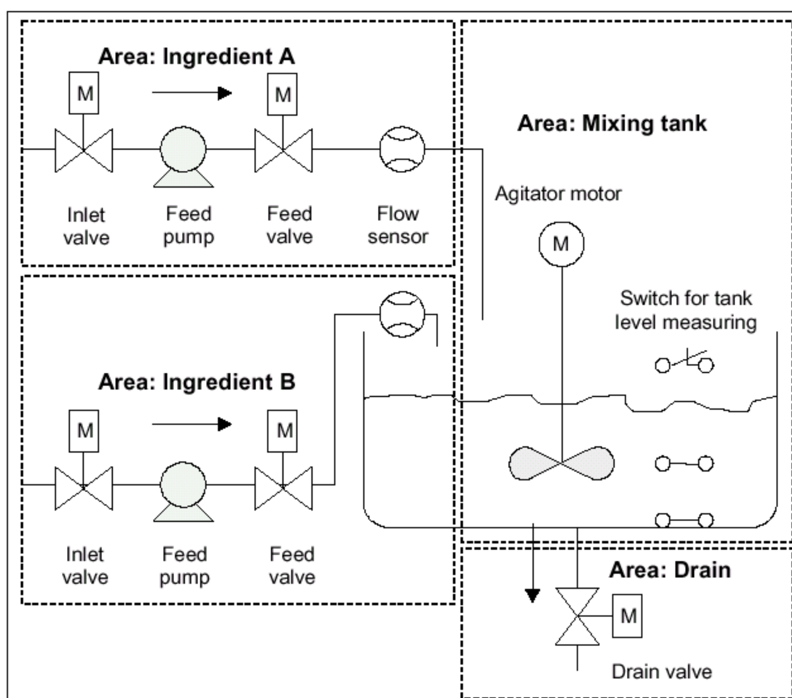
当安装 STEP 7 时，除非指定否则会复制提供的示例项目。如果已对提供的示例项目作了编辑，则这些修改过的项目在 STEP 7 重新安装时会被原版覆盖。为此，应该在作任何修改前复制提供的示例项目，然后在备份的项目上编辑。

A.5.2 工业搅拌过程的示例程序

本示例程序使用的关于控制一个工业搅拌过程的信息可能已在本手册的第一部中读过。

任务

两种配料(配料 A 和配料 B)在一个混合罐中由搅拌器混合在一起。混合后的产品通过一个排料阀排出该罐。下图为示例过程的框图。



过程中各个部分的描述

手册的第一部分包括了如何将示例过程分割为功能区域和单个任务的描述。下面将说明各个区域。

配料 A 区和配料 B 区：

- 每种配料的管道都配备有一个入口和一个进料阀以及进料泵。

- 进料管还有流量传感器。
- 当罐的液面传感器指示罐满时，进料泵的接通必须被锁定。
- 当排料阀打开时进料泵的启动必须被锁定。
- 在起动进料泵后最开始的1秒钟内必须打开入口阀和进料阀。
- 在进料泵停止后(来自流量传感器的信号)阀门必须立即被关闭以防止配料从泵中泄露。
- 进料泵的启动与一个时间监控功能相结合，换句话说，在泵启动后的7秒之内，流量传感器会报告溢出。
- 当进料泵运行时，如果流量传感器没有流量信号，进料泵必须尽可能快地断开。
- 进料泵启动的次数必须进行计数。(维护间隔)

混合罐区：

- 当罐的液面传感器指示“液面低于最低限”或排料阀打开时，搅拌电机的启动必须被锁定。
- 搅拌电机在达到额定速度时要发出一个响应信号。如果在电机启动后10秒内还未接收到该信号，则电机必须被断开。
- 必须对搅拌电机的启动次数进行计数(维护间隔)。
- 在混合罐中必须安装三个传感器；
    - 罐装满：一个常闭触点。当达到罐的最高液面时，该触点断开。
    - 罐中液面高于最低限：一个常开触点。如果达到最低限，该触点关闭。
    - 罐非空：一个常开触点，如果罐不空，该触点闭合。

排料区：

- 罐内产品的排出由一个螺线管阀门控制。
- 这个螺线管阀门由操作员控制，但是最迟在“罐空”信号产生时，该阀必须被关闭。
- 打开排料阀必须被锁定，当：
    - 搅拌电机在工作时
    - 罐空时

操作员站

要让一个操作员启动、停止和监控过程，就需要一个操作站。操作站有以下配备：

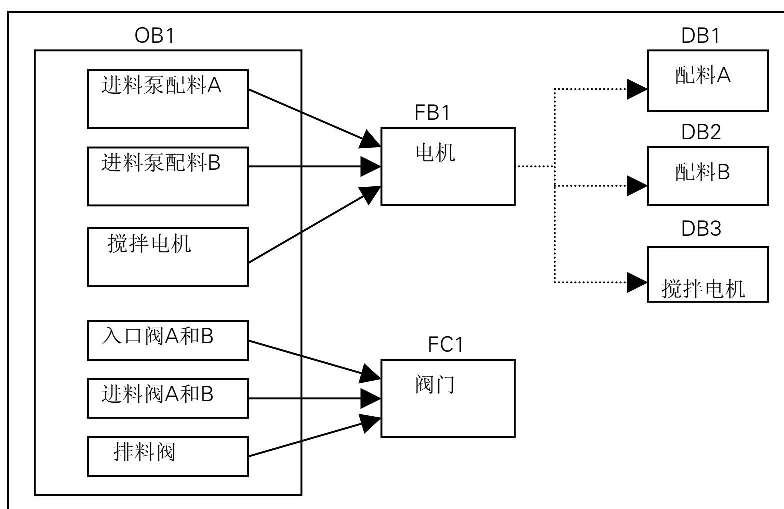
- 用于控制过程中最重要阶段的开关。使用“reset maintenance display(复位维护显示)”开关，可以关掉电机的维护显示灯，复位相应的计数器使维护间隔为0。
- 指示过程状态的显示灯。
- 紧急停机开关。

A.5.2.1 定义逻辑块

可以将用户程序分布到不同的块中并建立块调用的分层结构来结构化程序。

块调用的分层结构

下图所示为结构化编程的块的分层调用结构。



- OB1：与CPU操作系统的接口，包含主要程序。在OB1中调用块FB1和FC1并传送控制过程所需的特定参数。
- FB1：配料A的进料泵、配料B的进料泵和搅拌电机的控制由于要求一致(接通、断开、计数应用程序等)，所以可通过同一个功能块进行控制。
- 背景DB1-3：用于控制配料A、配料B的进料泵和搅拌电机的实参及静态数据各不相同，因此分别存储在与FB1相关的三个背景DB中。
- FC1：配料A和B的入口阀和进料阀以及排料阀也共同使用一个逻辑块。由于只需编辑“打开和闭合”功能，一个FC就足够了。

A.5.2.2 指定符号名

定义符号名

在示例程序中使用了符号，这些符号必须用 STEP 7 在符号表中定义。下表所示为所用程序组件的符号名及绝对地址。

进料泵、搅拌电机和入口阀的符号地址			
符号名	地址	数据类型	说明
Feed_pump_A_start	I0.0	BOOL	启动配料A的进料泵
Feed_pump_A_stop	I0.1	BOOL	停止配料A的进料泵
Flow_A	I0.2	BOOL	配料A流动
Inlet_valve_A	Q4.0	BOOL	启动配料A的入口阀
Feed_valve_A	Q4.1	BOOL	启动配料A的进料阀
Feed_pump_A_on	Q4.2	BOOL	“配料A的进料泵运行”指示灯
Feed_pump_A_off	Q4.3	BOOL	“配料A的进泵未运行”指示灯
Feed_pump_A	Q4.4	BOOL	配料A的进料泵启动
Feed_pump_A_fault	Q4.5	BOOL	“进料泵A故障”指示灯
Feed_pump_A_maint	Q4.6	BOOL	“进料泵A维护”指示灯
Feed_pump_B_start	I0.3	BOOL	启动配料B的进料泵
Feed_pump_B_stop	I0.4	BOOL	停止配料B的进料泵
Flow_B	I0.5	BOOL	配料B流动
Inlet_valve_B	Q5.0	BOOL	启动配料B的入口阀
Feed_valve_B	Q5.1	BOOL	启动配料B的进料阀
Feed_pump_B_on	Q5.2	BOOL	“配料B进料泵运行”指示灯
Feed_pump_B_off	Q5.3	BOOL	“配料B的进料泵未运行”指示灯
Feed_pump_B	Q5.4	BOOL	启动配料B的进料泵
Feed_pump_B_fault	Q5.5	BOOL	“进料泵B故障”指示灯
Feed_pump_B_maint	Q5.6	BOOL	“进料泵B维护”指示灯
Agitator_running	I1.0	BOOL	搅拌电机的响应信号
Agitator_start	I1.1	BOOL	搅拌启动按扭
Agitator_stop	I1.2	BOOL	搅拌停止按扭
Agitator	Q8.0	BOOL	启动搅拌器
Agitator_on	Q8.1	BOOL	“搅拌器运行”指示灯
Agitator_off	Q8.2	BOOL	“搅拌器未运行”指示灯
Agitator_fault	Q8.3	BOOL	“搅拌电机故障”指示灯
Agitator_maint	Q8.4	BOOL	“搅拌电机维护”指示灯

传感器和罐液面显示的符号地址			
符号名	地址	数据类型	说明
Tank_below_max	I1.3	BOOL	“混合罐未满”传感器
Tank_above_min	I1.4	BOOL	“混合罐液面高于最低限”传感器
Tank_not_empty	I1.5	BOOL	“混合罐非空”传感器
Tank_max_disp	Q9.0	BOOL	“混合罐满”指示灯
Tank_min_disp	Q9.1	BOOL	“混合罐液面低于低限”指示灯
Tank_empty_disp	Q9.2	BOOL	“混合罐空”指示灯

排料阀的符号地址			
符号名	地址	数据类型	说明
Drain_open	I0.6	BOOL	打开排料阀的按钮
Drain_closed	I0.7	BOOL	关闭排料阀的按钮
Drain	Q9.5	BOOL	启动排料阀
Drain_open_disp	Q9.6	BOOL	“排料阀打开”指示灯
Drain_closed_disp	Q9.7	BOOL	“排料阀关闭”指示灯

其它编程组件的符号地址			
符号名	地址	数据类型	说明
EMER_STOP_off	I1.6	BOOL	紧急停机开关
Reset_maint	I1.7	BOOL	复位所有电机上的维护指示灯的开关
Motor_block	FB1	FB1	控制泵和电机的FB
Valve_block	FC1	FC1	控制阀门的FC
DB_feed_pump_A	DB1	FB1	控制送料泵A的背景DB
DB_feed_pump_B	DB2	FB1	控制送料泵B的背景DB
DB_agitator	DB3	FB1	控制搅拌电机的背景DB

A.5.2.3 生成电机的 FB

该 FB 的要求是什么?

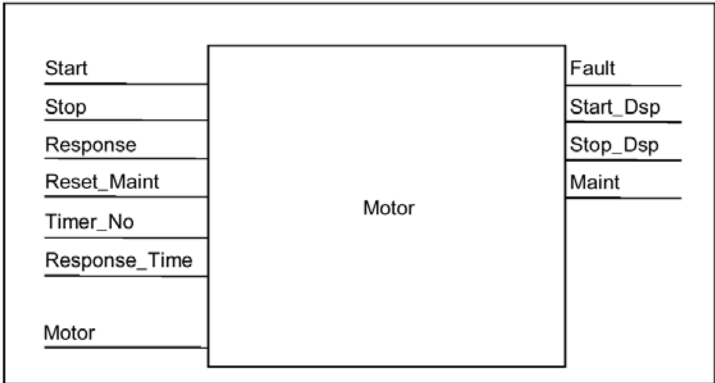
电机的 FB 包括以下逻辑功能：

- 有启动和停止输入。
- 允许设备操作的一系列互锁(泵和搅拌电机)。互锁状态存储在OB1的临时局域数据(L推栈)中(“Motor_enable”,“Valve-enable”)，并且当电机的FB被处理时与启动和停止输入进行逻辑组合。
- 来自设备的返馈必须在一个特定的时间内出现，否则就假定有故障或错误出现。该功能则停下电机。
- 时间点和响应时间或错误/故障循环持续时间都必须被指定。

- 如果启动按钮被按下并且电机被使能，设备自行接通并运行直至按下停机按钮。
- 当设备接通时，一个定时器启动运行，如果在定时器的时间到达之前未接到来自设备的响应信号，则停机。

指定输入和输出

下图所示是电机通用 FB 的输入和输出。



定义 FB 的参数

如果为电机(用于控制泵和电机)使用多重背景的 FB，必须为输入和输出定义通用参数名。

用于示例过程中的电机的 FB 需要以下各项：

- 它必须有来自操作站的信号可以停止或启动电机和泵。
- 它需要来自电机和泵的响应信号以指示电机在运行。
- 它必须计算从发出启动电机的信号到接收到响应信号的时间。如果在这一时间内没有收到响应信号则电机必须关断。
- 它必须能接通或断开操作站上的指示灯。
- 它提供一个信号启动电机。

这些要求可以被定义为 FB 的输入和输出。下表所示为我们的示例过程中电机的 FB 的参数。

参数名	输入	输出	入/出
Start	n		
Stop	n		
Response	n		
Reset_maint	n		
Timer_No	n		
Response_Time	n		
Fault		n	
Start_Dsp		n	
Stop_Dsp		n	
Maint		n	
Motor			n

为电机的 FB 声明变量

必须为电机的 FB 声明输入、输出和入/出参数。

地址	声明	变量名称	类型	初始值
0.0	IN	Start	BOOL	FALSE
0.1	IN	Stop	BOOL	FALSE
0.2	IN	Response	BOOL	FALSE
0.3	IN	Reset_Manit	BOOL	FALSE
2.0	IN	Timer_No	TIMER	
4.0	IN	Response_Time	S5TIME	S5T#0MS
6.0	OUT	Fault	BOOL	FALSE
6.1	OUT	Start_Dsp	BOOL	FALSE
6.2	OUT	Stop_Dsp	BOOL	FALSE
6.3	OUT	Maint	BOOL	FALSE
8.0	IN_OUT	Motor	BOOL	FALSE
10.0	STAT	Time_bin	WORD	W#16#0
12.0	STAT	Time_BCD	WORD	W#16#0
14.0	STAT	Starts	INT	0
16.0	STAT	Start_Edge	BOOL	FALSE

FB 的输入、输出、输入输/出和静态变量存储在指定的背景 DB 中，临时变量存储在 L 堆栈中。

为电机编程 FB

在 STEP 7 中，每一个被调用的块一定要在调用它的块之前生成。因此在示例程序中必须在 OB1 之前生成电机的 FB。

使用 STL 编程语言的 FB1 程序部分如下：

Network 1 启动/停止和锁存

```
A(
O    #Start
O    #Motor
)
AN   #Stop
=    #Motor
```

Network 2 启动监控

```
A    #Motor
L    #Response_Time
SD   #Timer_No
AN   #Motor
R    #Timer_No
L    #Timer_No
T    #Timer_bin
LC   #Timer_No
T    #Timer_BCD
```

```
          A     #Timer_No
          AN    #Response
          S     #Fault
          R     #Motor
Network 3 启动指示灯和故障复位
          A     #Response
          =     #Start_Dsp
          R     #Fault
Network 4 断开指示灯
          AN    #Response
          =     #Stop_Dsp
Network 5 启动计数
          A     #Motor
          FP    #Start_Edge
          JCN   lab1
          L     #Starts
          +     1
          T     #Starts
          lab1: NOP 0
Network 6 维护指示灯
          L     #Starts
          L     50
          >=I
          =     #Maint
Network 7 复位累计启动次数的计数器
          A     #Reset_Maint
          A     #Maint
          JCN   END
          L     0
          T     #Starts
          END:NOP 0
```

生成背景数据块

生成三个数据块并依次打打开。在“New Data Block(新数据块)”对话框中选择选项“Data block referencing a funtion block(参照一个功能块的数据块)”，在“Reference(参考)”列表框中选择“FB1”。则数据块被指定为 FB1 的背景数据块。

A.5.2.4 为阀门生成 FC

该 FC 的要求是什么？

这个入口和进料阀以及排料阀的功能包含以下逻辑功能：

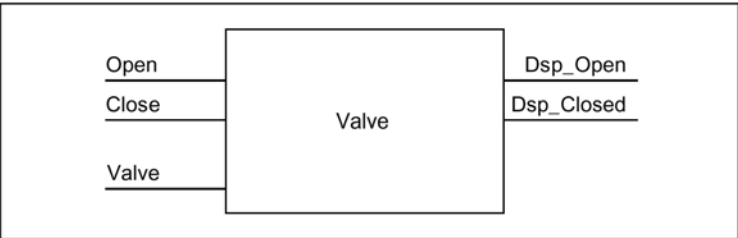
- 一个用于打开阀门的输入一个用于关闭门的输入。
- 互锁允许阀门被打开。互锁状态存储在OB1的临时局域数据(L堆栈)中(“Valve_enable”)并且在阀门的FC被处理时与打开和关闭的输入信号进行逻辑组合。

下表所示为必须传送给 FC 的参数。

阀门的参数	Input	Output	In/Out
Open	✓		
Close	✓		
Dsp_Open		✓	
Dsp_Closed		✓	
Valve			✓

指定输入和输出

下图所示为阀门的通用 FC 的输入和输出。调用 FB 为电机传送输入参数，调用 FC 为阀门返回输出参数。



为阀门的 FC 声明变量

就象为电机的 FB 一样，必须为阀门的 FC 声明输入、输出和输入/输出参数(见下面的变量声明表)

地址	声明	变量名称	类型	初始值
0.0	IN	Open	BOOL	FALSE
0.1	IN	Close	BOOL	FALSE
2.0	OUT	Dsp_Open	BOOL	FALSE
2.1	OUT	Dsp_Closed	BOOL	FALSE
4.0	IN_OUT	Valve	BOOL	FALSE

在 FC 中，临时变量存储在 L 堆栈。输入、输出和输入/输出变量则作为指针存储于调用 FC 的逻辑块中。L 堆栈中另外的存储空间(在临时变量之后)用于存储这些变量。

为阀门编程 FC

由于被调用的块必须在调用块之前生成，所以阀门的 FC1 功能必须在 OB1 之前生成。

使用 STL 编程语言的 FC1 程序部分如下：

Network 1 打开/关闭和锁存

```
A(
```

```
        O       #Open
        O       #Valve
        )
        AN      #Close
        =       #Valve
Netword 2 显示“阀门打开”
        A       #Valve
        =       #Dsp_Open
Network 3 显示“阀门关闭”
        AN      #Valve
        =       #Dsp_Closed
```

A.5.2.5 生成 OB1

OB1 决定示例程序的结构。OB1 中也包含要传送给各个功能的参数，例如：

- 为进料泵和搅拌电机而编制的STL程序段中为电机FB提供起动(“Start”)、停止(“Stop”)，响应(“Respense”)以及复位维护显示(“Reset_Maint”)的输入参数。PLC的每一个循环周期都会处理这个电机的FB。
- 如果电机的FB被处理，则输入Timer_No和Respense_Time指示所使用的定时器功能以及要测量的时间。
- 因为阀门的FC和电机的FB是在OB1中调用的，所以它们在每个程序循环都会被可编程控制器处理。

程序为处理进料泵和搅拌电机的控制任务，使用电机的 FB 时分别配备了不同的背景 DB。

为 OB1 声明变量

OB1 的变量声明表如下所示。前 20 个字节包含 OB1 的启动信息，不能够被修改。

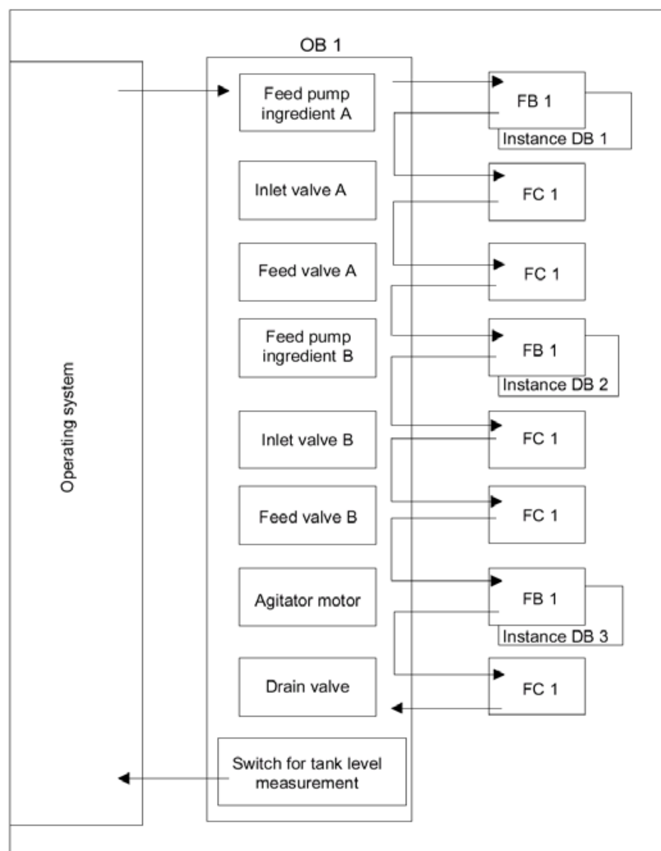
地址	变量声明	变量名	类型
0.0	TEMP	OB1_EV_CLASS	BYTE
1.0	TEMP	OB1_SCAN1	BYTE
2.0	TEMP	OB1_PRIORITY	BYTE
3.0	TEMP	OB1_OB_NUMBER	BYTE
4.0	TEMP	OB1_RESERVED_1	BYTE
5.0	TEMP	OB1_RESERVED_2	BYTE
6.0	TEMP	OB1_PREV_CYCLE	INT
8.0	TEMP	OB1_MIN_CYCLE	INT
10.0	TEMP	OB1_MAX_CYCLE	INT
12.0	TEMP	OB1_DATE_TIME	DATE_AND_TIME
20.0	TEMP	Enable_motor	BOOL
20.1	TEMP	Enable_valve	BOOL
20.2	TEMP	Start_fulfilled	BOOL

地址	变量声明	变量名	类型
20.3	TEMP	Stop_fulfilled	BOOL
20.4	TEMP	Inlet_valve_A_open	BOOL
20.5	TEMP	Inlet_valve_A_closed	BOOL
20.6	TEMP	Feed_valve_A_open	BOOL
20.7	TEMP	Feed_valve_A_closed	BOOL
21.0	TEMP	Inlet_valve_B_open	BOOL
21.1	TEMP	Inlet_valve_B_closed	BOOL
21.2	TEMP	Feed_valve_B_open	BOOL
21.3	TEMP	Feed_valve_B_closed	BOOL
21.4	TEMP	Open_drain	BOOL
21.5	TEMP	Close_drain	BOOL
21.6	TEMP	Valve_closed_fulfilled	BOOL

生成 OB1 程序

在 STEP 7 中，每个被调用的块必须在调用它的块之前生成。在示例程序中，必须在编程 OB1 之前生成电机的 FB 和阀门的 FC。FB1 和 FC1 在 OB1 中不止一次地被调用；FB1 的调用使用了不同的背景 DB：

参见下页框图。



STL 编程语言的 OBI 程序部分如下所示：

```
Network1  进料泵 A 的互锁
    A      “EMEP_STOP_off”
    A      “Tank_below_max”
    AN     “Driain”
    =      #Enable_Motor
Network2  为配料 A 调用电机的 FB
    A                “Feed_pump_A_start”
    A               #Enable_Motor
    =               #Start_Fulfilled
    A(
    O                “Feed_Pump_A_stop”
    ON              #Enable_Motor
    )
    =               #Stop_Fulfilled
    CALL  “Motor_block”,“DB_feed_pump_A”
          Start   :=Start_Fulfilled
          Stop    :=Stop_Fulfilled
          Response        :=“Flow_A”
          Reset_Maint     :=“Reset_maint”
          Timer_No        :=T12
          Reponse_Time    :=S5T#7S
          Fault   :=“Feed_pump_A_fault”
          Start_Dsp   :=“Feed_pump_A_on”
          Stop_Dsp    :=“Feed_pump_A_off”
          Maint :=“Feed_pump_A_maint”
          Motor :=“Feed_pump_A”
Network 3  延迟配料 A 的阀使能
    A      “Feed_pump_A”
    L      S5T#1S
    SD     T     13
    AN     “Feed_pump_A”
    R      T     13
    A      T     13
    =      #Enable_Valve
Network 4  配料 A 的入口阀控制
    AN     “Flow_A”
    AN     “Feed_pump_A”
    =            #Close_Valve_Fulfilled
    CALL  “Valve_baock”
            Open       :=#Enable_Valve
            Close      :=#Close_Valve_Fulfilled
            Dsp_Open       :=#Inlet_Valve_A_Open
```

```
            Dsp_Closed      :=#Inlet_Valve_A_Closed
            Valve        :=“Inet_Valve_A”
Network 5  配料 A 的进料阀控制
    AN    “Flow_A”
    AN    “Feed_pump_A”
    =     #Close_Valve_Fulfilled
    CALL  “Valve_block”
          Open         :=#Enable_Valeve
          Close        :=#Close_Valve_Fulfilled
          Dsp_Open  :=#Feed_Valve_A_Open
          Dsp_Closed:=#Feed_Valve_A_Closed
          Valve  :=“Feed_Valve_A”
Network 6  进料泵 B 的互锁
    A     “EMER_STOP_off”
    A     “Tank_below_max”
    AN    “Drain”
    =     “Enable_Motor”
Network 7  为配料 B 调用电机的 FB
    A                    “Feed_pump_B_start”
    A                    #Enable_Motor
    =                    #Start_Fulfilled
    A(
    O                    “Feed_pump_B_stop”
    ON                   #Enable_Motor”
    )
    =        #Stop_Fulfilled
    CALL                 “Motor_block”,“DB_feed_pump_B”
          Star    :=#Start_Fulfilled
          Stop    :=#Stop_Fullfilled
          Response   :=“Flow_B”
          Reset_Maint      :=“Reset_maint”
          Timer_No    :T14
          Reponse_Time    :=S5T#7S
          Fault   :=“Feed_pump_B_fault”
          Start_Dsp   :=“Feed_pump_B_on”
          Stop_Dsp    :=“Feed_pump_B_off”
          Maint :=“Feed_pump_B_mait”
          Motor :=“Feed_pump_B”
Network 8  延迟配料 B 的阀使能
    A     “Feed_pump_B”
    L     S5T#1S
    SD    T      15
    AN    “Feed_pump_B”
```

```
    R     T     15
    A     T     15
    =     #Enable_Valve
Network 9  配料 B 的入口阀控制
    AN    “Flow_B”
    AN    “Feed_pump_B”
    =     #Close_Valve_Fulfilled
    CALL  “Valve-block”
          Open  :=#Enable_Valve
          Close  :=#Close_Valve_Fulfilled
          Dsp_Open  :=#Inlet_Valve_B_Open
          Dsp_Closed:=#Inlet_Valve_B_Closed
          Valve  :=“Inlet_Valve_B”
Network 10  配料 B 的进料阀控制
    AN    “Flow_B”
    AN    “Feed_pump_B”
    =     #Close_Valve_Fulfilled
    CALL  “Valve_block”
          Open  :=#Enable_Valve
          Close  :=#Close_Valve_Fulfilled
          Dsp_Open  :=#Feed_Valve_B_Open
          Dsp_Closed:=#Feed_Valve_B_Closed
          Valve  :=“Feed_Valve_B”
Network 11  搅拌器的互锁
    A     “EMER_STOP_off”
    A     “Tabj_above_min”
    AN    “Drain”
    =     #Enable_Motor
Network 12  为搅拌器调用电机的 FB
    A     “Agitator_Start”
    A     #Enable_Motor
    =     #Start_Fulfilled
    A(
    O     “Agitator_stop”
    ON    #Engable_Motor
    )
    =     #Stop_Fulfilled
    CALL  “Motor_block”,“DB_Agitator”
          Start   :=#Start_Fulfilled
          Stop    :=#Stop_Fullfilled
          Response   :=“Agitator_running”
          Reset_Maint      :=“Reset_maint”
          Timer_No   :=T16
```

```
            Reponse_Time    :=S5T#10S
            Fault   := “Agitator fault”
            Start_Dsp   := “Agitator on”
            Stop_Dsp   := “Agitator_off”
            Maint  := “Agitator_maint”
            Motor := “Agitator”
Network 13  排料阀的互锁
    A       “EMER_STOP_off”
    A       “Tank_not_empty”
    AN      “Agitator”
    =       “Enable_Valve”
Network 14  排料阀控制
    A       “Drain_open”
    A       #Enable_Valve
    =       #Open_Drain
    A(
    O       “Drain_closed”
    ON      #Enable_Valve
    )
    =       #Close_Drain
    CALL    “Valve_baock”
            Open  :=#Open_Drain
            Close  :=#Close_Drain
            Dsp_Open  := “Drain_open_disp”
            Dsp_Closed       := “Drain_closed_disp”
            Valve  := “Drain”
Network 15Tank level display 罐液面显示
    AN      “Tank_below_max”
    =       “Tank_max disp”
    AN      “Tank_above_min”
    =       “Tank_min_disp”
    AN      “Tank_not-empty”
    =       “Tank_empty_disp”
```

A.5.3 处理日时钟中断举例

“日时钟中断” 的用户程序的结构

FC 12

OB 10

OB1 和 OB80

A.5.3.1 “日时钟中断”的用户程序的结构

任务

输出 Q4.0 要在周一的上午 5：00 至周五的下午 8：00 被置位，从周五的下午 8：00 至周一的上午 5：00 被复位。

转换为用户程序

下表所示为每个块的子任务

块	分任务
OB1	调用功能FC12
FC12	视输出Q4.0的状态、日时钟中断的状态及输入I0.0和I0.1而定 • 指定起始时间 • 设置日时钟中断 • 激活日时钟中断 • CAN_TINT
OB10	视当前的星期而定 • 指定起始时间 • 置位或复位输出Q4.0 • 设置下一个日时钟中断 • 激活下一个日时钟中断
OB80	置位输出Q4.1 在位存储区域保存OB80的起动信息

使用的地址

下表所示为使用的共享地址。临时局域变量在各个块的声明区内声明。

地址	含义
I0.0	使能“设置日时钟中断”和“激活日时钟中断”的输入
I0.1	取消一个日时钟中断的输入
Q4.0	由日时钟中断OB（OB10）置位/复位的输出
Q4.1	由时间错误（OB80）置位的输出
MW16	日时钟中断的状态（SFC31”QRY_TINT”）
MB100至MB107	OB10(只有日时钟)的起动事件信息存储区
MB110至MB129	OB80(时间错误)的坡动事件信息存储区
MW200	SFC 28”SET_TINT”的RET_VAL(返回值)
MB202	SFC的二进制结果（状态位BR）缓存区
MW204	SFC 30”ACT_TINT”的RET_VAL
MW208	SFC 31”QRY_TINT”的RET_VAL

使用的系统功能和功能

在编程示例中使用了以下系统功能：

- SFC28“SET_TINT”：设置日时钟中断
- SFC29“CAN_TINT”：取消日时钟中断
- SFC30“ACT_TINT”：激活日时钟中断
- SFC31“QRY_TINT”：查询日时钟中断
- SFCT“D_TOD_DT”：组合DATE和TIME_OF_DAY为DT。

A.5.3.2 FC12

声明部分

以下是在 FC12 的声明部分声明的临时局域变量：

Variable Name	DateType	Declaration	Comment
IN_TIME	TIME_OF_DAY	TEMP	起动时间
IN_DATE	DATE	TEMP	起动日期
OUT_TIME_DATE	DATE_AND_TIME	TEMP	转换后的起动日期/时间
OK_MEMORY	BOOL	TEMP	使能日时钟中断设置

STL 程序部分

使用 STL 编写的用户程序 FC12 如下：

STL(FC12)	解释
Network 1 CALL SFC 31 OB_NO := 10 RET_VAL:= MW 208 STATUS := MW 16	SFC QRY_TINT 查询日时钟中断的状态
Network 2 AN Q 4.0 JC mond L D#1995-1-27 T #IN_DATE L TOD#20:0:0.0 T #IN_TIME JU CNVRT mond:L D#1995-1-23 T #IN_DATE	依据 Q4.0 指定起始时间（在变量 #IN_DATE 和#IN_TIME 中） 起始日期是星期五 起始日期是星期一

<table>
<tr><td>

```
        L     TOD#5:0:0.0
        T     #IN_TIME
cnvrt:NOP  0
Network 3:
        CALL  FC  3
          IN1     := #IN_DATE
          In2     := #IN_TIME
          RET_VAL:= #OUT_TIME_DATE
Network 4:
        A     I 0.0
        AN    M 17.2
        A     M 17.4
        =     #OK_MEMROY
Network 5:
        A     #OK_MEMORY
        JNB   m001
        CALL  SFC 28
          OB_NO   := 10
          SDT     := #OUT_TIME_DATE
          PERIOD  := W#16#1201
          RET_VAL := MW 200
m001 A      BR
     =      M 202.3
Network 6:
        A     #OK_MEMMORY
        JNB   m002
        CALL SFC 30
          OB_N := 10
          RET_VAL := MW 204
m002  A     BR
      =     M202.4
Network 7:
       A      I0.1
       JNB    M003
       Call    SFC29
       OB_NO:=10
       RET_VAL:=MW210
m003:  A     BR
        =     M202.5
```

</td><td>
从 DATE 和 TIME_OF_DAY 转换格式为 DATE_AND_TIME(用于设置日时钟中断)

设置日时钟中断的要求是否全部满足？（使能输入置位、日时钟中断未激活以及日时钟中断 OB 已装载）如果满足，设置日时钟中断…

…并且激活日时钟中断

如果取消日时钟中断的输入置位了，则取消日钟中断
</td></tr>
</table>

A.5.3.3 OB10

声明部分

与 OB10 的缺省声明部分不同，以下临时局域变量要进行声明：

- 用于整个起动事件信息的结构体(STARTINFO)
- 在STARTINFO结构体内用于时间的结构体(T_STMP)
- 其它临时局域变量WDAY，IN_DATE、IN_TIME和OUT_TIME_DATE

变量名	数据类型	变量声明	注释
STARTINFO	STRUCT	TEMP	OB10的整个起动事件信息声明为结构体
E_ID	WORD	TEMP	事件ID
PR_CLASS	BYTE	TEMP	优先级
OB_NO	BYTE	TEMP	OB号码
RESERVED_1	BYTE	TEMP	保留
RESERVED_2	BYTE	TEMP	保留
PERIOD	WORD	TEMP	日时钟中断的周期
RESERVED_3	DWORD	TEMP	保留
T_STMP	STRUCT	TEMP	用于日时钟明细数据的结构体
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	
WDAY	INT	TEMP	星期
IN_DATE	DATE	TEMP	FC3的输入变量 （日期转换格式）
IN_TIME	TIME_OF_DAY	TEMP	FC3的输入变量 （时间转换格式）
OUT_TIME_DATE	DATE_AND_TIME	TEMP	FC3的输出变量和SFC28的输入变量

STL 程序部分

使用 STL 编写的用户程序 OB10 如下：

STL(OB10)	解释
Network 1	
L #STARTINFO.7_STMP.MSEC_WDAY	选择星期
L W#16#F	
AW	
T #WDAY	并存储
Network 2:	
L #WDAY	如果不是周一,则指定周一,5.00AM为下一个
L 2	起动时间并复位输出Q4.0
<>I	
JC mond	
Network 3:	
L D#1995-1-27	否则，如果是周一，指定周五，8.00pm (20.00)
T #IN_DATE	作为下一个起动时间并置位输出Q4.0
L TOD#20:0:0.0	
T #IN_TIME	
SET	
= Q 4.0	
JU cnvrt	
mond: L D#1995-1-23	
T #IN_DATE	
L TOD#5:0:0.0	
T #IN_TIME	
CLR	
= Q 4.0	启动指定的时间
	转换指定的起始时间为DATE_AND_TIME格
cnvrt: NOP 0	式(用于SFC28)
Network 4:	
CALL FC 3	
IN1 := #IN_DATE	
IN2 := #IN_TIME	
RET_VAL := #OUT_TIME_DATE	
Network 5:	
CALL SFC 28	
OB_NO := 10	
SDT := #OUT_TIME_DATE	
PERIOD := W#16#1201	
RET_VAL := MW 200	
A BR	
= M 202.1	
Network 6:	
CALL SFC 30	激活日时钟中断
OB_NO := 10	
RET_VAL := MW 204	

A BR = M 202.2 Network 7: CALL SFC 20 SRCBLK := #STARTINFO.T_STMP RET_VAL := MW 206 DSTBLK := P#M 100.0 BYTE 8	块传送:将OB10的起动事件信息中的日时钟存储到存储区域MB100至MB107

A.5.3.4 OB1 和 OB80

由于在本例中没有评估 OB1(用于循环程序的 OB)的起动事件信息，所以只显示 OB80 的起动事件信息。

OB1 程序部分

使用 STL 编写的用户程序 OB1 如下：

STL(OB1)	解释
CALL FC12	调用功能FC12

OB80 声明部分

与 OB80 的缺省声明部分不同，以下临时局域变量要进行声明：

- 用于整个起动事件信息的结构体(STARTINFO)
- 在STARTINFO结构体内的用于时间的结构体(T_STMP)

变量名	数据类型	变量声明	注释
STARTINFO	STRUCT	TEMP	OB80的整个起动事件信息声明为结构体
E_ID	WORD	TEMP	事件ID
PR_CLASS	BYTE	TEMP	优先级
OB_NO	BYTE	TEMP	OB号码
RESERVED_1	BYTE	TEMP	保留
RESERVED_2	BYTE	TEMP	保留
A1_INFO	WORD	TEMP	关于造成错误的事件的附加信息
A2_INFO	DWORD	TEMP	关于事件优先级和故障OB号码的附加信息
T_STMP	STRUCT	TEMP	用于日时钟明细数据的结构体
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	

变量名	数据类型	变量声明	注释
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

OB80 程序部分

使用 STL 编写的用户程序 OB10 如下，如果出现时间错误，操作系统会调用 OB80：

STL(OB80)	解释
Network 1 AN Q 4.1 S Q 4.1	如果时间错误出现，置位Q4.1
CALL SFC 20 SRCBLK := #STARTINFO RET_VAL := MW 210 DSTBLK := P#M 110.0 BYTE 20	块传送：将整个起动事件信息存储到存储区域MB110至MB129

A.5.4 处理时间延迟中断的示例

“时间延迟中断”的用户程序结构

OB20、OB1

A.5.4.1 “时间延迟中断”用户程序的结构

任务

当输入 I0.0 置位，输出 Q4.0 应在 10 秒后被置位。每次输入 I0.0 置位，延迟时间都将被启动。

时间延迟中断的起动时间(秒和毫秒)应作为用户定义的 ID 出现在时间延迟中断 OB(OB20)的起动事件信息中。

如果在这 10 秒钟内 I0.1 被置位，则组织块 OB20 不应被调用；这意味着输出 Q4.0 不应被置位。

当输入 I0.2 置位时，输出 Q4.0 应被复位。

转换为用户程序

下表为每个块的子任务。

块	分任务
OB1	读当前时间并为起动时间延迟中断作准备 根据输入I0.0的边沿变化，起动时间延迟中断；根据时间延迟中断的状态以及输入I0.1的边沿变化，取消时间延迟中断；要据输入I0.2的状态，复位输出Q4.0
OB20	置位输出Q4.0 读并且准备当前时间 将起动事件信息存到位存储区域

使用的地址

下表所示为使用的共享地址，临时局域变量在各个块的声明区域内声明。

地址	含义
I0.0	使能“起动时间延迟中断”的输入
I0.1	取消一个时间延迟中间的输入
I0.2	复位输出Q4.0的输入
Q4.0	由时间延迟中断OB(OB20)置位的输入
MB1	用作边沿标志及SFC的二进制结果（状态位BR）缓存
MW4	时间延迟中断的状态（SFC34”QRY_TINT”）
MD10	来自OB1起动事件信息的秒和毫秒的BCD码
MW100	SFC32”SRT_DINT”的RET_VAL（返回值）
MW102	SFC34”QRY_DINT”的RET_VAL
MW104	SFC33”CAN_DINT”的RET_VAL
MW106	SFC20”BLKMOV”的RET_VAL
MB120-MB139	OB20的起动事件信息的存储区
MD140	来自OB20的起动事件信息的秒和毫秒的BCD码
MW144	来自OB1的起动事件信息的秒和毫秒的BCD码；取自OB20的起动事件信息（用户指定IDSIGN）

使用的系统功能

以下是在“时间延迟中断”的用户程序中使用的 SFC：

- SFC32“SRT_DINT”：启动时间延迟中断
- SFC33“CAN_DINT”：取消时间延迟中断
- SFC34“QRY_DINT”：查询时间延迟中断的状态

A.5.4.2 OB20

声明部分

与 OB20 的缺省声明部分不同，下列临时局域变量要进行声明：

- 用于整个起动事件信息的结构体(STARTINFO)
- 在STARTINFO结构体中用于时间的结构体(T_STMP)

变量名	数据类型	变量声明	注释
STARTINFO	STRUCT	TEMP	OB20的起动信息
E_ID	WORD	TEMP	事件ID
PC_NO	BYTE	TEMP	优先级
OB_NO	BYTE	TEMP	OB号码
D_ID1	BYTE	TEMP	数据ID1
D_ID2	BYTE	TEMP	数据ID2
SIGN	WORD	TEMP	用户指定的ID
DTIME	TIME	TEMP	时间延迟中断的起动时间
T_STMP	STRUCT	TEMP	用于日时钟明细数据的结构体（时间印记）
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

程序部分

使用 STL 编写的用户程序 OB20 如下：

STL(OB20)	解释
Network 1	
SET	无条件置位输出Q4.0
= Q 4.0	
Network 2:	
L QW 4	立即启动输出字
T PQW 4	
Network 3:	
L #STARTINFO.T_STMP.SECONDS	从起动事件信息中读秒
T MW 140	
L #STARTINFO.T_STMP.MSEC_WDAY	从起动事件信息中读毫秒和星期
T MW 142	

L MD 140 SRD 4 T MD 140	去掉星期并将毫秒重新写回（现在MW142中为BCD码）
Network 4: L #STARTINFO.SIGN T MW 144	从起动事件信息中读时间延迟中断的起动时间（=调用SFC32）
Network 5: CALL SFC 20 SRCBLK := STARTINFO RET_VAL := MW 106 DSTBLK := P#M 120.0 Byte 20	将起动事件信息复制到存储区域（MB120至MB139）

A.5.4.3 OB1

声明部分

与 OB1 中缺省声明部分不同，以下临时局域变量要进行声明：

- 用于整个起动事件信息的结构体(STARTINFO)
- 在STARTINFO结构体中用于时间的结构体(T_STMP)

变量名	数据类型	变量声明	注释
STARTINFO	STRUCT	TEMP	OB1的起动信息
E_ID	WORD	TEMP	事件ID
PC_NO	BYTE	TEMP	优先级
OB_NO	BYTE	TEMP	OB号码
D_ID1	BYTE	TEMP	数据ID1
D_ID2	BYTE	TEMP	数据ID2
CUR_CYC	INT	TEMP	当前循环时间
MIN_CYC	INT	TEMP	最小循环时间
MAX_CYC	INT	TEMP	最大循环时间
T_STMP	STRUCT	TEMP	用于日时钟明细数据的结构体（时间印记）
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

程序部分

使用 STL 编写的用户程序 OB10 如下：

STL(OB1)	解释
Network 1:	
L #STARTINFO.T_STMP.SECONDS	从起动事件信息中读秒值
T MW 10	
L #STARTINFO.T_STMP.MSEC_WDAY	从起动事件信息中读毫秒和星期
T MW 12	去掉星期并重新写回毫米（现为BCD码在MW12中）
L MD 10	
SRD 4	
T MD 10	
Network 2:	输入I0.0是否有上升沿？
A I 0.0	
FP M 1.0	
= M 1.1	
Network 3:	
A M 1.1	如果有，起动时间延迟中断
JNB m001	（时间延迟中断的起始时间赋给参数SIGN）
CALL SFC 32	
OB_NO := 20	
DTME := T#10S	
SIGN := MW 12	
RET_VAL :=MW 100	
m001: NOP 0	
Network 4:	查询时间延迟中断的状态
CALL SFC 34	(SFC QRY_DINT)
OB_NO := 20	
RET_VAL := MW 102	
STATUS := MW 4	输入I0.1有上升沿吗？
Network 5:	
A I 0.1	
FP M 1.3	
= M 1.4	
Network 6:	…并且时间延迟中断已被激活吗
A M 1.4	（时间延迟中断状态的位2）？
A M 5.2	则取消时间延迟中断
JNB m002	
CALL SFC 13	
OB_NO := 20	
RET_VAL :=104	
m002: NOP 0	
A I 0.2	用输入I0.2复位输出Q4.0
R Q 4.0	

A.5.4.4 屏蔽和中断屏蔽同步错误的示例

以下用户程序的示例说明如何屏蔽和中断屏蔽同步错误。使用 SFC36“MSK_FLT”，以下错误可以在编程错误筛选器中被屏蔽：

- 作读操作时的区域长度错误
- 作写操作时的区域长度错误

再次调用 SFC36“MSK_FLT”也可以屏蔽一个访问区域：

- 作写操作时I/O访问错误

使用 SFC38“READ-ERR”可查询被屏蔽的同步错误。用 SFC37“DMSK-FLT”“写操作时 I/O 访问错误”的屏蔽可以被取消。

程序部分

在下面的 OB1 中可以看到使用语句表编程的用户程序示例。

STL(Network 1)	解释
AN M 255.0	非保持存储位M255.0(只有第一次运行时=0)
JNB m001 CALL SFC 36	SFC 36 MSK_FLT(屏蔽同步错误) 位2=位3=1(BLFL和BLFS被屏蔽)
PRGFLT_SET_MASK :=DW#16#C	所有位=0(没有访问错误被屏蔽)
ACCFLT_SET_MASK :=DW#16#0	返回值
RET_VAL :=MW 100 PRGFLT_MASKED :=MD 10	输出当前编程错误筛选器到MD10
	输出当前编程错误筛选器到MD14
ACCFLT_MASKED :=MD14	
	如果屏蔽成功置位M255.0
m001: A BR S M 255.0	

STL(Network 2)	解释
CALL SFC 36	SFC 36 MSK_FLT(屏蔽同步错误)
PRGFLT_SET_MASK :=DW#16#0	所有位=0(没有更进一步的编程错误被屏蔽)
ACCFLT_SET_MASK :=DW#16#8	位3=1(写操作访问错误被屏蔽) 返回值
RET_VAL :=MW 102 PRGFLT_MASKED :=MD 20	输出当前编程错误筛选器到MD20
ACCFLT_MASKED :=MD 24	输出当前编程错误筛选器到MD24

STL(Network 3)	解释
AN M 27.3 BEC	如果写访问错误(ACCFLT_MASKED中的位3)没有屏蔽则块结束

STL(Network 4)	解释
L B#16#0	
T PQB 16	向PQB16作写访问(用数值0)

STL(Network 5)	解释
CALL SFC 38	SFC 38 READ_ERR(查询同步错误) 所有位=0(没有编程错误被查询)
PRGFLT_QUERY :=DW#16#0	位3=1(查询写访问错误)
ACCFLT_QUERY :=DW#16#8	返回值
RET_VAL :=MW 104 PRGFLT_CLR :=MD 30	输出当前编程错误筛选器到MD 30
ACCFLT_CLR :=MD 34	输出当前编程错误筛选器到MD 34
A BR	无错误出现并且在检查写访问错误
A M 37.3 NOT = M 0.0	RLO取反 如果PQB16存在则M0.0=1

STL(Network 6)	解释
L B#16#0 T PQB 17	向PQB17作写访问(用数值0)

STL(Network 7)	解释
CALL SFC 38	SFC 38 READ_ERR(查询同步错误) 所有位=0(没有编程错误被查询)
PRGFLT_QUERY :=DW#16#0	位3=1(查询写访问错误)
ACCFLT_QUERY :=DW#16#8	返回值
RET_VAL :=MW 104 PRGFLT_CLR :=MD 30	输出当前编程错误筛选器到MD 30
	输出当前编程错误筛选器到MD 34
ACCFLT_CLR :=MD 34	
	无错误出现并且在检查写访问错误
A BR A M 37.3 NOT = M 0.0	RLO取反 如果PQB17存在则M0.1=1

STL(Network 8)	解释
L B#16#0 T PQB 18	向PQB18作写访问(用数值0)

STL(Network 9)	解释
CALL SFC 38	SFC 38 READ_ERR(查询同步错误)
PRGFLT_QUERY :=DW#16#0	所有位=0(没有编程错误被查询)
ACCFLT_QUERY :=DW#16#8	位3=1(查询写访问错误)
RET_VAL :=MW 104	返回值
PRGFLT_CLR :=MD 30	输出当前编程错误筛选器到MD 30
ACCFLT_CLR :=MD 34	输出当前编程错误筛选器到MD 34
A BR	无错误出现并且在检查写访问错误
A M 37.3	
NOT	RLQ取反
= M 0.2	如果PQB18存在则M0.2=1

STL(Network 10)	解释
L B#16#0	向PQB19作写访问(用数值0)
T PQB 19	

STL(Network 11)	解释
CALL SFC 38	SFC 38 READ_ERR(查询同步错误)
PRGFLT_QUERY :=DW#16#0	所有位=0(没有编程错误被查询)
ACCFLT_QUERY :=DW#16#8	位3=1(查询写访问错误)
RET_VAL :=MW 104	返回值
PRGFLT_CLR :=MD 30	输出当前编程错误筛选器到MD 30
ACCFLT_CLR :=MD 34	输出当前编程错误筛选器到MD 34
A BR	无错误出现并且在检查写访问错误
A M 37.3	
NOT	RLQ取反
= M 0.3	如果PQB19存在则M0.3=1

STL(Network 12)	解释
CALL SFC 37	SFC 37 READ_ERR(中断屏蔽同步错误)
PRGFLT_QUERY :=DW#16#0	所有位=0(没有进一步的编程错误被屏蔽)
ACCFLT_QUERY :=DW#16#8	位3=1(屏蔽写访问错误)
RET_VAL :=MW 102	返回值
PRGFLT_CLR :=MD 20	输出当前编程错误筛选器到MD 20
ACCFLT_CLR :=MD 24	输出当前编程错误筛选器到MD 24

STL(Network 13)	解释
A M 27.3 BEC	如果写访问错误(ACCFLT_MASKED位3)未被中断屏蔽则块结束

STL(Network 14)	解释
A M 0.0 JNB m002 L IB 0 T PQB 16 m002: NOP 0	如果存在，传送IB0到PQB16

STL(Network 15)	解释
A M 0.1 JNB m003 L IB 1 T PQB 17 m003: NOP 0	如果存在，传送IB1到PQB17

STL(Network 16)	解释
A M 0.2 JNB m004 L IB 2 T PQB 18 m004: NOP 0	如果存在，传送IB2到PQB18

STL(Network 17)	解释
A M 0.3 JNB m005 L IB 3 T PQB 19 m005: NOP 0	如果存在，传送IB3到PQB19

A.5.4.5 禁止和使能中断及异步错误的示例(SFC39 和 SFC40)

在这个用户程序的示例中，假设某个程序部分不能被中断中断。为这部分程序使用 SFC39 “DIS_IRT” 禁止 OB35(循环中断)的调用，以后可以用 SFC40 “EN-IRT” 再重新使能。

SFC39 和 SFC40 在 OB1 中调用：

STL(OB1)	解释
A M 0.0 S M 90.1 A M 0.1 S M 90.0 : :	可以被中断的程序部分：
CALL SFC 39 MODE :=B#16#2 OB_NO :=35 RET_VAL :=MW 100 : :	不能被中断的程序部分： 禁止和放弃中断 模式2:禁止单个中断OB 禁止OB35
L PIW 100 T MW 200 L MW 90 T MW 92 : :	
CALL SFC 40 MODE :=B#26#2 OB_NO :=35 RET_VAL :=MW 102	使能中断 模式2：使能单个中断OB 使能OB35
A M 10.0 S M 190.1 A M 10.1 S M 190.0 : :	可以被中断的程序部分

A.5.4.6 中断和异步错误的延迟处理示例(SFC41 和 SFC42)

在本例的用户程序中，假定某部分程序不能被中断中断。为这部分程序使用 SFC41 “DIS_AIRT” 使中断延迟，以后再用 SFC42 “EN_AIRT” 使能。

SFC41 和 SFC42 在 OB1 中调用：

STL(OB1)	解释
A M 0.0	可以被中断的程序部分：
S M 90.1	
A M 0.1	
S M 90.0	
:	
:	
CALL SFC 41	不能被中断的程序部分：
RET_VAL :=MW 100	禁止和延迟中断
L PIW 100	
T MW 200	
L MW 90	
T MW 92	
:	
:	
CALL SFC 42	
RET_VAL :=MW 102	使能中断
L MW 100	设置中断禁止的次数在返回值中
DEC 1	该数值在中断使能后应与中断禁止前有相同的值（这里为“0”）
L MW 102	
<>I	
JC err	
A M 10.0	
S M 190.1	
A M 10.1	
S M 190.0	可以被中断的程序部分
:	
:	
BEU	
err: L MW 102	
T QW 12	显示已设置的中断禁止的次数

A.6 访问过程数据区和外设数据区

A.6.1 访问过程数据区

CPU 既可以通过使用过程映像表间接访问中央或分布式数字输入/输出模板的输入和输出，也可以通过背板/P 总线直接访问。

CPU 通过背板/P 总线直接访问中央输入和输出和分布式模拟输入/输出模板。也可以为模拟量模块分配过程映像区地址。

模板询址

可按如下所示，在用 STEP 7 组态模板时，将用户程序中使用的地址分配给模板：

- 对于中央的I/O模板：在组态表中布置机架并将模板分配到各个槽。
- 对于带有分布式I/O(PROFIBUS-DP)的站：在组态表的“master system(在系统)”中安排DP从站，为其设置PROFIBUS地址并将模板分配到各槽。

通过组态模板，不再需要用开关在每个模板上设置地址。作为组态的结果，编程器把数据发送到 CPU 使得 CPU 能够识别分配给它的模板。

外设 I/O 寻址

输入和输出有不同的地址区域。这意味着外设区域的地址不仅要包括访问类型字节、字，而且还要有输入的标识符 I 和输出的标识符 Q。

下图所示为可用的外设地址区域。

地址区域	访问单元	S7标识符（IEC）
外设（I/O）区域：输入	外设输入字节 外设输入字 外设输入双字	PIB PIW PID
外设（I/O）区域：输出	外设输出字节 外设输出字 外设输出双字	PQB PQW PQD

要查找在各个模板上可以使用哪些地址区域，可参考以下手册：

- “S7-300可编程控制器，硬件和安装”手册
- “S7-300，M7-300可编程控制器，模板技术规范参考手册
- “S7-400，M7-400可编程控制器，模板技术规范参考手册

模板起始地址

模板起始地址是该模板的最低字节地址。它代表该模板的用户数据区域的起始地址，在许多情况下用来代表整个模板。

例如，在硬件中断、诊断中断、插入/移出模板中断以及电源故障中断中都要在相应的组织块的启动信息中输入模板的起始地址，用以识别引起中断的模板。

A.6.2 寻址外设数据区

外设数据区域可被分为以下几部分：

- 用户数据以及
- 诊断和参数数据

两个区域都有输入区域(只读)和输出区域(只写)。

用户数据

用户数据可以用字节地址(对数字信号模板)或字地址(对模拟量模板)来寻址其输入或输出区域。用户数据可以用装载和传送命令、通过功能(接作员接口访问)来访问，或者通过传送过程映像区。用户数据可以是以下各项：

- 来自信号模板的数字和模拟输入/输出信号
- 来自功能模板的控制和状态信息
- 来自通讯模板的点到点和总线连接信息(只有S7-300)

当传送用户数据时，最高可传送 4 个字节(DP 标准从站除外，见设置操作性能)。如果使用“传送双字”指令，四个连续且不可修改(一致的)字节被传送。如果你分别用四个“传送输入字节”指令，则硬件中断 OB 可能会插在指令之间以及插在向同一地址传送数据之间，这样 4 个字节后来的内容在它们被全部传送之前可能会被改变。

诊断和参数数据

模板的诊断和参数数据不能够被单独寻址但总可以以一套完整的数据被传送，这意味着总可以传送一致的诊断和参数数据。

对诊断和参数数据的访问可以使用模板的起始地址和数据记录号(DS)。数据记录可分为输入和输出数据记录。输入数据记录只能读，输出数据记录只能写。可以用系统功能或通讯功能(用户接口)访问数据记录。下表所示为数据记录与诊断和参数数据之间的关系。

数据	说明
诊断数据	如果该模板有诊断能力，则可通过读数据记录0和1获得模板的诊断数据
参数数据	如果该模板可组态，则可通过向数据记录0和1作写操作而将参数传送到模板

访问数据记录

可以使用模板的数据记录中的信息给可组态模板重新赋值参数，并且可以读有诊断能力的模板的诊断信息。

下表所示为哪些系统功能可以用来访问数据记录

SFC	目的
给模板赋值参数	
SFC55 WR_PARM	传送可修改参数（数据记录1）到寻址的信号模板
SFC56 WR_DPARM	从SDB100至129传送参数（数组0或1）到寻址的信号模板
SFC57 PARM_MOD	从SDB100至129传送参数（数组0或1）到寻址的信号模板
SFC58 WR_REC	传送任意数据到寻址的信号模板
读出诊断信息	
SFC59 RD_REC	读诊数据数据

注意：

如果一个 DPV1 从站被组态为使用 GSD 文件（GSD Rev.3），并且 DP 主站的 DP 接口设置为“S7 兼容通讯”，用户程序通过 SFC58/59 或 SFB53/52 不能访问 I/O 模板的数据纪录。原因是此时 DP 主站对插槽询址错误（组态的插槽+3）。

解决方法：将 DP 主站的接口设置到“DPV1”。

寻址 S5 模板

可以按如下所述访问 S5 模板：

- 使用接口模板IM463-2将一个S7-400与SIMATIC S5扩展机架相连
- 将装在适配盒中的S5模板插在S7-400的中央机架上。

如何在 SIMATIC S7 中寻址 S5 模板的解释在“S7-400，M7-400 可编程控制器，硬件和安装”手册中或随适配盒提供的说明中。

A.7 设置操作性

本章将解释如何通过设置系统参数或使用系统功能(SFC)修改 S7-300 和 S7-400 可编程控制器的某些特性。

可以在 STEP 7 的在线帮助中以及下列手册中找到关于模板参数的更详细信息：

- “S7-300可编程控制器，硬件和安装”手册
- “S7-300，M7-300可编程控制器，模板技术规范”参考手册
- “S7-400，M7-400可编程控制器、模板技术规范”参考手册

在“S7-300 和 S7-400 系统软件，系统和标准功能”参考手册中可以找到关于 SFC 的所有内容。

寻址 DP 标准从站

如果与 DP 标准从站之间的数据交换大于 4 个字节，则必须使用用于这种数据交换的特殊的 SFC。

一些新的 CPU 中通过 I/O 地址区交换大于 4 个字节的连续数据时，不再需要调用 SFC14/SFC15（参考对分布式 I/O 连续数据读写的描述）。

SFC	目的
给模板赋值参数	
SFC 15 DPWR_DAT	传送任意数据到寻址的信号模板
读出诊断信息	
SFC 13 DPNRM_DG	读诊断信息（异步读访问）
SFC 14 DPRD_DAT	读连续的数据（长度为3个字节或大于4个字节）

当一个 DP 诊断帧到达时，一个有 4 个字节诊断数据触发 CPU 诊断中断。可以调用 SFC13 DPNRM_DG 读出这 4 个字节。

A.7.1 改变模板的性能和特性

缺省设置

- 出厂时，所有S7可编程控制器的可组态模板都有适合与标准应用的缺省设置。用这些缺省值，可以立即使用模板而无需作任何设置。这些缺省值的解释在以下手册的模板说明中：
- “S7-300可编程控制器，硬件和安装”手册
- “S7-300、M7-300可编程控制器，模板技术规范”参考手册
- “S7-400，M7-400可编程控制器，模板技术规范”参考手册

哪些模板可以赋值参数?

可以修改模板的性能和特性，使它们适应当前的要求和工厂情况。可组态的模板是 CPU、FM、CP 以及一些模拟量输入/输出模板和数字量输入模板。

可组态模板有的有后备电池，有的没有后备电池。

没有后备电池的模板在任何掉电之后都必须重新提供数据。这些模板的参数保存在 CPU 的可保持存储区域中(由 CPU 作间接的参数赋值)。

设置和装载参数

可以用 STEP 7 设置模板参数。当存储这些参数时，STEP 7 生成对象“系统数据块”，它与用户程序一起下载到 CPU，当 CPU 启动时被传送到模板上。

可以作哪些设置?

模板参数被分为参数块。在“S7-300 可编程控制器，硬件和安装”手册以及“S7-400，M7-400 可编程控制器，模板技术规范”参考手册中解释了哪些参数块在哪些 CPU 上有效的。

参数块举例：

- 启动性能
- 循环
- MPI
- 诊断
- 可保持数据
- 时钟存储
- 中断处理

- 集成I/O(只有S7-300)
- 保护等级
- 局域数据
- 实时时钟
- 异步错误

用 SFC 作参数赋值

除了用 STEP 7 作参数赋值外，还可以用 S7 程序的系统功能修改模板参数。下表说明哪些 SFC 传送哪些模板参数。

SFC	目的
SFC 55 WR_PARM	传送可修改的参数（数据记录1）到寻址的信号模板
SFC 56 WR_DPAPM	从相应的SDB传送参数（数据记录0或1）到寻址的信号模板
SFC 57 PARM_MOD	从相应的SDB传送所有参数（数据记录0或1）到寻址的信号模板
SFC 58 WR_REC	传送任意数据记录到寻址的信号模板

在“S7-300 和 S7-400 系统软件，系统和标准功能”参考手册中有关于系统功能的详细描述。

以下手册中解释了哪些模板参数可以被动态修改：

- “S7-300可编程控制器，硬件和安装”手册
- “S7-300，M7-300可编程控制器，模板技术规范”参考手册
- “S7-400，M7-400可编程控制器，模板技术规范”参考手册

A.7.2 离线更新模块或子模块的固件版本（操作系统）

按下列步骤向存储器卡传送更新文件：

1. 建立一个新的路径
2. 从更新软盘上将 UPD 文件拷贝到该目录
3. 选择菜单命令 **PLC > Update Operating System**
4. 在出现的对话框中选择 UPD 文件的目录
5. 选择任何一个 UPD 文件
6. 点击“OK”，退出对话框

将存储器卡插入 PLC 中。

执行操作系统的更新

1. 关掉 CPU 的电源(PS 模板)
2. 将准备好的装有操作系统更新文件的存储器卡插入 PLC 中
3. CPU 上电。从存储器卡中将操作系统传送到内部 FLASH-EPROM 中，操作期间，CPU

上的所有指示灯点亮

4. 大约两分钟后，操作系统完成更新。CPU 上的 STOP 指示灯通过慢闪来指示更新的完成(系统请求存储器复位)
5. 关掉电源，插入想要运行的存储器卡
6. 接通电源，CPU 将自动执行存储器复位，之后 CPU 就可以运行了。
7. 更新其它模块和子模块的固件版本。

A.7.3 使用时钟功能

所有的 S7-300/-400CPU 都配备有一个时钟(实时时钟或软件时钟)。这个时钟可在可编程控制器中，即能作主时钟也能作具有被外步同步的从时钟。日时钟中断和 CPU 运行时间计数器数器需要这个时钟。

时间格式

时钟总是指示时间(最小精度 1 秒)、日期和星期。有些 CPU 还可以指示毫秒(参考“S7-300 可编程控制器，硬件和安装”手册以及“S7-400，M7-400 可编程控制器模板技术规范”参考手册)。

设置和读时间

可以在用户程序中通过调用 SFC0 SET_CLK 设置 CPU 时间和日期时钟，或者在编程器上使用菜单选项启动时钟。使用 SFC1 READ_CLK 或编程器上的菜单选项可以读 CPU 当前的日期和时间。

为时钟赋值参数

如果在一个网络中不止一个模板配有时钟，必须用 STEP 7 设置参数指定当时间同步时哪个 CPU 功能为主，哪个为从。当设置参数时，还要决定时间是通过通讯总线被同步还是通过多点接口，以及对时间作自动同步的间隔。

同步时间

为确保网络中的所有模板的时间是一样的，从时钟则由系统程序以一个定期的(可选)间隔对它同步，也可以调用系统功能 SFC48 SFC_RTCB 将主时钟的日期和时间传送给从时钟。

使用一个 CPU 运行时间计数器数器

CPU 运行时间计数器对所连接的设备的操作时间或者 CPU 的总的运行时间进行计数。

在 STOP 模式，运行时间计数器数器停止，它的计数值在存储器复位后仍可保持。在暖起动期间，CPU 运行时间计数器必须被用户程序重新启动；在热起动期间，如果它以前就已被起动了，现在则可自动运行。

可以用 SFC2 SET_RTM 给 CPU 运行时间计数器设置一个初始值。可以用 SFC 3

CTRL_RTM 启动或停止 CPU 运行时间计数器。可以用 SFC4 READ_RTM 读当前总的操作小时数和计数器的状态(“停止”或“计数”)。

一个 CPU 最多可有八个 CPU 运行时间计数器。号码从 0 开始。

A.7.4 使用时钟存储器和定时器

时钟存储器

时钟存储器信号存储在位存储器的一个字节中，每个位的状态按照 1：1 的比例周期性地为 1 和为 0。当使用 STEP 7 为时钟存储器赋值参数时，可以选择在 CPU 上使用哪个位存储器字节。

用途

可以在用户程序中使用时钟存储器字节，例如，启动一个闪烁灯或触发一个周期性的操作(如，测量一个实际值)。

可能的频率

时钟字节的每一位被赋予一个频率。下表为赋值关系

时钟存储字节的位	7	6	5	4	3	2	1	0
周期(s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
频率(Hz)	0.5	0.625	1	1.25	2	2.5	5	10

提示

时钟存储字节与 CPU 循环不同步，也就是说，在一个长的周期内，时钟存储字节的状态可能会改变若干次。

定时器

定时器是系统存储器的一个存储区域。可以在用户程序中指定一个定时器功能(如延迟接通定时器)。可用的定时器的数据依据 CPU 而定。

提示

- 如果在用户程序中使用的定时器的数量超过了CPU所允许的，则出现同步错误且触发OB121启动。
- 在S7-300上(CPU318除外)，只能在OB1和OB100中同时启动并更新定时器，在其它OB中定时器只能被启动。